National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

DREA CR/97/459

CLASSIFICATION OF SIMULATES SEQUENCES USING A **FIR** NEURAL NETWORK: EXPERIMENTAL RESULTS

Normand Arsenault — Maryhelen Stevenson

Center for Research in Engineering and Applied Science
University of New Brunswick
P.O.Box 440, Fredricton, N.B., Canada
E3B 5A3

CONTRACTOR REPORT

Prepared for

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

106-03650/

THIS IS AN UNEDITED REPORT ON SCIENTIFIC OR TECHNICAL WORK CONTRACTED BY THE DEFENCE RESEARCH ESTABLISHMENT ATLANTIC OF THE RESEARCH AND DEVELOPMENT BRANCH OF THE DEPARTMENT OF NATIONAL DEFENCE, CANADA.

THE CONTENTS OF THE REPORT ARE THE RESPONSIBILITY OF THE CONTRACTOR, AND DO NOT NECESSARILY REFLECT THE OFFICIAL POLICIES OF THE DEPARTMENT OF NATIONAL DEFENCE.

PLEASE DIRECT ENQUIRIES TO:

THE DIRECTOR GENERAL
DEFENCE RESEARCH ESTABLISHMENT ATLANTIC
P.O. BOX 1012
DARTMOUTH, NOVA SCOTIA, CANADA
B2Y 3Z7

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

DREA CR/97/459

CLASSIFICATION OF SIMULATES SEQUENCES USING A **FIR** NEURAL NETWORK: EXPERIMENTAL RESULTS

Normand Arsenault — Maryhelen Stevenson

Center for Research in Engineering and Applied Science
University of New Brunswick
P.O.Box 440, Fredricton, N.B., Canada
E3B 5A3

Accepted by : J.N. Maksym - Contract Scientific Authority

W7707-3-2682/01-OSC
Contract Number

April 1995

CONTRACTOR REPORT

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

Abstract

This report examines the experimental results achieved by the FIR Neural Network on data sets which simulate underwater acoustic transient signals. A software package to simulate FIR neural networks has been developed in the C programming language. The effects of several user controlled options will be given along with suggestions on how to choose these and the network parameters (number of layers, number of nodes, *etc.*) which define the structure of the FIR neural network. Experiments have been conducted using three separate data sets (each data set contains a training and test set composed of a collection of spectrograms obtained from the Short Time Fourier Transform): a Three Tone Data Set (TTDS), the Simulated DARPA Data Set (SDDS), and a MultiPath Data Set (MPDS). The effects of four user controlled options (weight initialization, weight update frequency, choice of desired response, and error scaling sequence) on the classification rate and learning speed is studied in order to determine the best configuration in which to set up the FIR software package. A suggested procedure to determine the transient time of occurrence is presented along with some results obtained on the SDDS. The performance of the FIR neural network is shown to be superior to that of static neural networks on both the TTDS and the SDDS. Finally, conclusions and suggestions for future work are given.

Résumé

Le rapport examine les résultats expérimentaux obtenus par le réseau neuronal FIR pour des ensembles de données qui simulent des signaux transitoires acoustiques sous-marins. Un logiciel de simulation de réseaux neuronaux FIR a été préparé en langage de programmation C. Les effets de plusieurs options déterminées par l'utilisateur sont présentés avec des suggestions sur la façon de choisir celles-ci et les paramètres de réseau (nombre de couches, nombre de noeuds, etc.) qui définissent la structure du réseau neuronal FIR. Des expériences ont été menées avec trois ensembles de données distincts (chaque ensemble de données contient un ensemble de formation et un ensemble d'essai, établis à partir d'un groupe de spectrogrammes obtenus de la transformée de Fourier rapide) : un ensemble de données à trois tonalités (TTDS), un ensemble de données DARPA simulé (SDDS) et un ensemble de données à trajets multiples (MPDS). Les effets de quatre options déterminées par l'utilisateur (initialisation du poids, fréquence de mise à jour du poids, choix de la réponse désirée et séquence de mise à l'échelle d'erreur) sur le taux de classification et la vitesse d'apprentissage sont étudiés pour établir la meilleure configuration du logiciel FIR. Une procédure suggérée visant à déterminer le temps d'occurrence transitoire est exposée avec certains résultats obtenus du SDDS. La performance du réseau neuronal FIR s'avère supérieure à celle des réseaux neuronaux statiques, tant pour le TTDS que pour le SDDS. Enfin, des conclusions et suggestions de travaux futurs sont présentées.

TABLE OF CONTENTS

Abstract ii

Table of Contents iii

1 Introduction 1

2 Software Set–Up and Performance Assessment 3

2.1 Software Features 3

2.2 Choosing Network Parameters 3

2.3 Network Parameter Notation 5

2.4 Measures of Performance 6

3 Data Set Description 8

3.1 Three–Tone Data Set 8

3.2 Simulated DARPA Data Set (SDDS) 14

3.3 Multipath Data Set (MPDS) 22

4 Weight Initialization and Update Frequency 28

4.1 Weight Initialization 28

4.2 Weight Update Frequency 29

4.3 Summary 31

5 Desired Response and Error Scaling 32

5.1 Results 33

5.2 Summary 35

6 Time of Occurrence 37

6.1 Method of Approach 37

6.2 Observations for Determining Transient Alignment 46

6.3 Suggested Procedure for Determining Transient Alignment 46

7 Comparison with Static Neural Networks 49

7.1 Three Tone Data Set 49

7.2 Simulated DARPA Data Set 50

7.3 Summary 53

8 Behavior in Multipath Environment **54**

9 Conclusions and Suggestions for Future Work **56**

9.1 Suggestions for Future Work 58

9.2 Test on Actual Data 58

9.3 Continuous Mode Operation 58

9.4 Experiment with other types of preprocessing techniques 59

9.5 Further experimentation with error–scaling and desired–response sequences 59

9.6 Further experimentation with architecture 59

9.7 Further experimentation with learning factors and momentum 59

9.8 Optimize the FIRNN code for more efficient operation 60

9.9 Hybrid System 60

9.10 Behavior in a Multipath environment 60

References **61**

1 Introduction

Various challenges associated with the task of processing and classifying underwater acoustic transient signals are discussed in [2] and include the following:

- the transients are highly nonstationary,
- the transients are impulsive in nature,
- there is significant intra–class variation due to changes in SNR, propagation channel and multipath effects, and
- the received signal may contain two or more events (transients originating from different sources) overlapped in time.

Due to the challenges mentioned above, the classification of acoustic transients is a difficult problem. Many solutions to this problem have been proposed. Recently numerous studies have suggested the use of Artificial Neural Networks (ANNs) as a means of classifying acoustic information. The FIR Neural Network architecture is well–suited to temporal processing tasks and was thus chosen for the purposes of this contract.

Numerous experiments have been conducted to determine

- the effects of various training options on the classification rate; these options include: the frequency of weight updates, various types of desired responses, and various error scaling techniques,
- the transients time of occurrence,
- the ability of the FIR network to classify signals with significant intra class variations; these include: variations due to changes in the multipath channel model, variations in SNR, etc.
- how the performance of the FIR network compares with static neural networks.

Determination of the FIR Neural Network's ability to classify acoustic transients required three major steps. The first step was to develop the software to implement the FIR structure, this was done in the C programming language running on a Unix based Sun Sparc–Station. A user's manual for the software package has also been developed as part of this contract. The underlying philosophy of the FIR neural network, the temporal backpropagation algorithm used to train the FIR network, and the relationship between FIR neural networks and static neural networks are discussed in yet another report [10] (also generated as part of this contract). The second step in the experimentation phase was to generate data sets which would accurately model the features of acoustic transient signals. To

date, three distinct data sets have been created. The first, termed the "*Three–Tone Data Set*" (TTDS) is based on three tones placed in various configurations with numerous intra–class variations. The second data set, the "*Simulated DARPA Data Set*" (SDDS) was based on the data set supplied by DARPA, the Defense Advanced Research Projects Agency of the United States to several researchers in that country as part of the Artificial Neural Networks initiative to test the effectiveness of several different neural network architectures[1]. For further reference to research performed with the DARPA data set the reader may wish to examine the work of Kundu *et al.* [1], Ghosh *et al.* [2], and Hemminger and Pao [3]. The work of these three groups is summarized in [10]. The third data set, "*Multipath Data Set*" (MPDS) was generated in an attempt to simulate acoustic transients with intra–class variations due to changes in a multipath channel. The third step in the experimentation process consisted of training and testing the network, and gathering the results for analysis and/or presentation. Result analysis and presentation was done in Matlab, Version 4.1 but any software package capable of data manipulation and graphing should be adequate.

The generation of multiple data sets stemmed from the desire to test the FIR network on problems of greater difficulty and realism. Hopefully the result is a software package which has evolved to a point where it is flexible, efficient and most importantly effective when tested with actual underwater acoustic information.

1. SDDS was developed and generated at UNB from the descriptions mentioned in [1, 2] and contains additions and modifications to the original DARPA data set.

2 Software Set-Up and Performance Assessment

Before proceeding to a description of the data sets and their subsequent results, we will comment briefly on: the features of the software, choice of network parameters, the notation used to refer to these parameters, and the techniques used to assess the performance of the network.

2.1 Software Features

The FIR Neural Network software package is set up in a non-interactive fashion; all the inputs required by the network are contained in, or referenced from, the user options file (for the exact format requirements of this file see [11]). The non-interactive approach was chosen because entering this information repeatedly is very time consuming and error prone. The contents of this file allow the user to control several features related to the training and testing of the network. The options allow the user to:

- specify the length of training,
- specify how the initial weights are assigned,
- decide the frequency of evaluating the classification accuracy on the test and training set,
- decide the frequency of evaluating the mean square error for the test and training set,
- save the updated weight values,
- specify the type of desired response,
- specify the type of error scaling,
- specify frequency of weight updates,
- save the output from the final or penultimate layer, and
- vary the network architecture by modifying yet another file which contains the network parameters.

2.2 Choosing Network Parameters

A very important factor in the success of an implementation will be the choice of network parameters which include; the number of layers, number of nodes per layer, filter order for each layer of FIR connections[2], and the learning factor for each layer. As many as possible of these parameters should be set from *a priori* knowledge of the problem.

2.2.1 Choosing the Number of Layers

Although there are no processing units and no adaptable weights associated with the network inputs, the software package includes the inputs to the network as a layer (layer num-

ber 0). As discussed in sections 1.4.1 and 2.2.1 of the User's Manual [11], the weights of the input layer units are nonadaptable and, in fact, the output units are not fully connected to the penultimate layer units. Instead, each unit in the output layer is connected only to the corresponding unit in the penultimate layer, and the weights of the output layer units are all set equal to a constant (with the bias weights of these units being set to zero). The philosophy behind this architecture is discussed in section 4.1.1 of [10]. Thus, an $(L+1)$ layer network (with layer numbers 0, ... , L) has only L layers of processing units (layers 1, ... , L). All the experimental results reported in this document were obtained using a four layer network, (*i.e.*, a network with two layers of adaptable weights). Such a network is capable of implementing arbitrarily complex decision surface [12] and has been successfully applied to phoneme classification tasks [4].

2.2.2 Choosing the Number of Nodes in Each Layer

The number of nodes in the input layer must agree with the dimensionality of the input sequence (our input sequences are spectrograms, the dimensionality of which is determined by the number of frequency bins). The number of nodes in both the penultimate layer and the output layer must be set equal to the number of possible pattern classes. The number of nodes in the intermediate layers is dependant upon the complexity of the input patterns and is usually determined empirically; in general, the number of nodes in intermediate layers will be greater than the number of output classes.

2.2.3 Choosing the Filter Order Between Each Layer

All filter connections to nodes in a particular layer, ℓ, have the same filter order, $M^{(\ell)}$. The filter order for the entire FIR neural network is defined as:

$$M^{(tot)} = \sum_{\ell=1}^{L} M^{(\ell)}. \tag{1}$$

The filter order of the entire network should be set equal to one less than the number of time frames associated with each input sequence or spectrogram. The optimal assignment of filter orders to intermediate layers is determined somewhat empirically. *A priori* knowledge of the pattern classes may also influence this assignment (e.g., see sections 4.1.1 and

2. The software assumes that all connections to nodes on a given layer have the same filter order; however, the filter order can vary from layer to layer.

4.1.2 of [10]). The filter order of the final layer, $M^{(L)}$ was usually set to a value equal to the possible misalignment (expressed in number of time frames) of the transient within the input sequence.

2.2.4 Choosing the Learning Factor for Each Layer

The software allows each layer, ℓ, of the network to be assigned its own learning factor, $\mu^{(\ell)}$. All the processing units within a particular layer are assigned the same learning factor. For the work presented in this report, we have found a learning factor of 0.001 (for all layers) to be a good initial choice. If the initial choice of learning factor appears to be too low or too high, we recommend changing it by a factor of two. Too high a learning rate will often reveal itself when plotting the classification rate versus training epoch, the curve will appear erratic or noisy, too low a learning rate will appear as a very slow improvement in classification rate versus training epoch. In some cases, it is desirable to start training with a large learning factor and then decrease the learning factor as training proceeds. If desired, this can be accomplished using the "save weights" option.

Although the software allows for assigning different learning factors to different layers, we have not experimented with this option.

Due to the large number of variables and extended range over which they vary, choosing the network parameters can be a time consuming process. Often different architectures will yield similar results after training for long periods. In general, some comfort may be taken in the fact that numerous "reasonable" architectures will eventually converge to similar classification rates.

2.3 Network Parameter Notation

In order to describe the architecture of a FIR network the following notation is introduced. Let FIR[$(N^{(0)}:0)$ x $(N^{(1)}:M^{(1)})$ x ... x $(N^{(L)}:M^{(L)})$] denote the architecture of an FIR network with L layers of processing units, where $N^{(0)}$ denotes the number of inputs; $N^{(1)}$, ... , $N^{(L)}$ denote the number of nodes in layers 1 to L respectively; and $M^{(1)}$, ... , $M^{(L)}$ denote the filter orders of the connections to the processing units of layers 1 through L. The filter order of the input layer is by definition set to zero since no processing units are associated

with this layer. Thus, to denote a FIR network with: 16 inputs; 10, 8, and 8 nodes in layers 1 to 3; and filter orders of 4, 6, and 2 for layers 1 to 3, we would write FIR[(16:0) x (10:8) x (8:6) x (8:2)]. Let $\mu^{(0)}$, ... , $\mu^{(L)}$ represent the learning factor for layers 0 through L, $\mu^{(0)}$ is also by definition set to zero since no processing units are associated with this layer. In all our experiments we have used a uniform learning rate (*i.e.*, equal learning rate for all the layers with processing units).

2.4 Measures of Performance

The software package includes two functions which can be used to assess the performance of the FIR neural network; these include a function to evaluate the mean square error (between desired output and actual output) on a specified data set, and a function to evaluate the classification rate of a specified data set.

2.4.1 Classification Rate

The classification rate is defined as the number of correct pattern classifications divided by the total number of patterns in the data set multiplied by 100. A correct classification occurs when the final layer node with the highest output corresponds with the desired class which is supplied to the network. Currently, this check is only performed on the T^{th} iteration, where T is the number of time frames per input sequence (for this work the input sequence is a spectrogram). The user has the option of choosing a detailed presentation of the classification results, or a summary of the classification results. In the case that a detailed version is selected, classification results for each pattern in the set are written to a file; this is useful for generating confusion matrices.

2.4.2 Mean Square Error

The mean square error is computed over the entire pattern set (training or test set), during which the weights of the network are held constant. The MSE is computed as the weighted sum of square errors over iterations 0 to $2T$–1, over all the nodes in the final layer, and

over all the spectrograms in the pattern set.

$$\text{MSE} = \left[\frac{1}{P}\sum_{p=1}^{P}\left[\frac{1}{N^{(L)}}\sum_{j=1}^{N^{(L)}}\left(\frac{1}{2T-1}\sum_{i=1}^{2T-1}\text{ErrorWeight}(i) * \text{error}^2(i)\right)\right]\right] \qquad (2)$$

The weighted sum of squared errors is the cost function which the training procedure attempts to minimize with respect to the patterns in the training set. To evaluate the cost function, each pattern is clocked in and out of the network, while holding the weights constant; at each instant of time, the error between the desired and actual network output is computed, squared and weighted by the appropriate component of the ErrorWeight array in the case that the user has chosen to invoke the error scaling option.

3 Data Set Description

As mentioned, three distinct data sets have been created, the following sections will describe the models used to generate the various signal sets. In all cases, feature extraction consists of generating a spectrogram representation for each signal. The spectrogram shows the energy distribution of the signal over time and frequency.

3.1 Three-Tone Data Set

The training set and test set each contain 999 patterns, representing three different pattern classes. The patterns in the training and testing sets were generated as follows. First, the class of each pattern was chosen randomly for a uniform distribution over the possible pattern classes: 0, 1, and 2. This results in a testing and training set with approximately equal numbers of patterns from all classes. Once the pattern class was determined, the appropriate class model was used to generate a sampled time signal. The three signal classes were modelled as follows:

Class 1

$$S_1(k) \equiv S_1(kT_s) = A_1 e^{-\alpha_1(k-k_1)T_s} \sin(2\pi f_1 kT_s) + A_2 e^{-\alpha_2(k-k_2)T_s} \sin(2\pi f_2 kT_s) + n(k) \quad (3)$$

Class 2

$$S_2(k) \equiv S_2(kT_s) = A_1 e^{-\alpha_1(k-k_1)T_s} \sin(2\pi f_2 kT_s) + A_2 e^{-\alpha_2(k-k_2)T_s} \sin(2\pi f_1 kT_s) + n(k) \quad (4)$$

Class 3

$$S_3(k) \equiv S_3(kT_s) = A_3 e^{-\alpha_3(k-k_3)T_s} \sin(2\pi f_3 kT_s) + n(k) \quad (5)$$

where k ranges between 1 and 152, the sampling period, T_s is 1 ms. and the random variables A_1, A_2, A_3, α_1, α_2, α_3, k_1, k_2, k_3, f_1, f_2, and f_3 determine the amplitudes, widths, locations, and frequencies of the simulated signals. Each of these variables is uniformly distributed over the range of values shown in Table 1. The additive noise sequence, $n(k)$ was simulated using a gaussian random number generator with mean 0 and variance 1. The possible alignments of the generated signals are determined by the random variables k_1, k_2, and k_3. Classes 1 and 2 each consist of two sinusoidal pulses. The location of the first pulse is determined by the random variable k_1 which can vary over 51 samples (this corresponds to 6 time frames in the spectrogram representation). The location of the second pulse is dependant on the location of the first pulse; the distance between pulses varies

between 20 and 70 samples. The absolute location of the second pulse may vary over a range of 101 samples (this corresponds to 13 time frames of the spectrogram representation). The alignment of class 3 signals is determined by the random variable k_3 and can vary over a range of 81 samples or 10 spectrogram time frames.

Table 1: Variation Range for Signal Variables in the Three Tone Data Set

Variable/s	Range
A_1, A_2, and A_3	1.0 – 1.5
α_1, α_2, and α_3	70 – 120
k_1	20 – 70
k_2	(k_1+20) – (k_1+70)
k_3	20 – 100
f_1	150 – 200 Hz.
f_2	350 – 400 Hz.
f_3	150 – 400 Hz.

Each signal consists of 152 points (*i.e.*, $k=1$ to 152), representing a time duration of 152 ms. For spectrogram generation, each signal was divided into 16 overlapping time frames of 32 points each; the overlap between consecutive time frames was 75%. A 32 point Hamming Window was applied to each time frame and the STFT was computed. This process resulted in a spectrogram which consisted of 16 spectral coefficients for each of the 16 time frames (or 256 elements per spectrogram). Each spectrogram was then normalized by its peak value resulting in normalized spectrograms with values ranging between 0 and 1. Figure 1 shows noise–free samples of the three signal classes along with their respective spectrograms. Figures 2, 3, and 4 show a collection of noise–free samples from Class 1, Class 2, and Class 3 respectively, in order to demonstrate the intra–class variations.

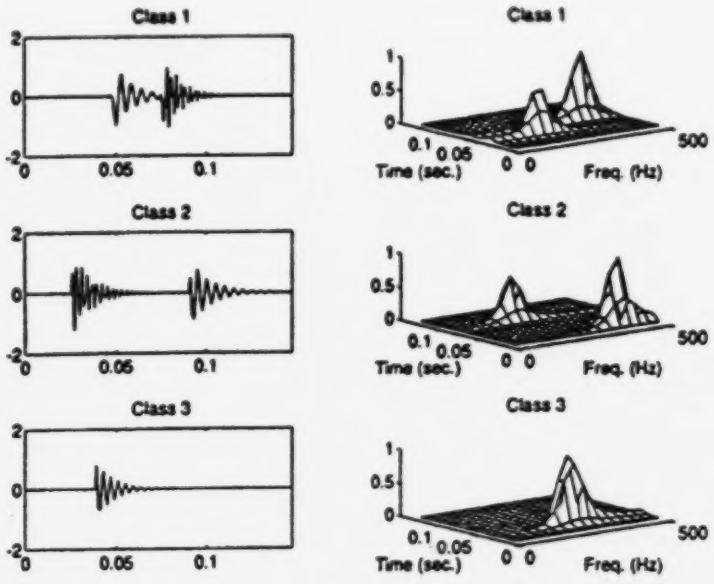

Figure 1: Samples of noise–free signals for Classes 1, 2 and 3 for the Three Tone Data Set.

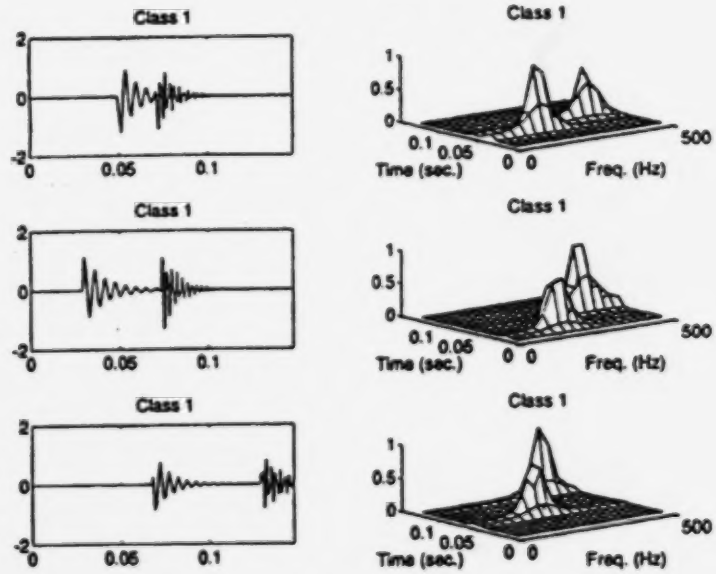

Figure 2: Noise–free Class 1 signals from the Three Tone Data Set with their respective spectrograms.

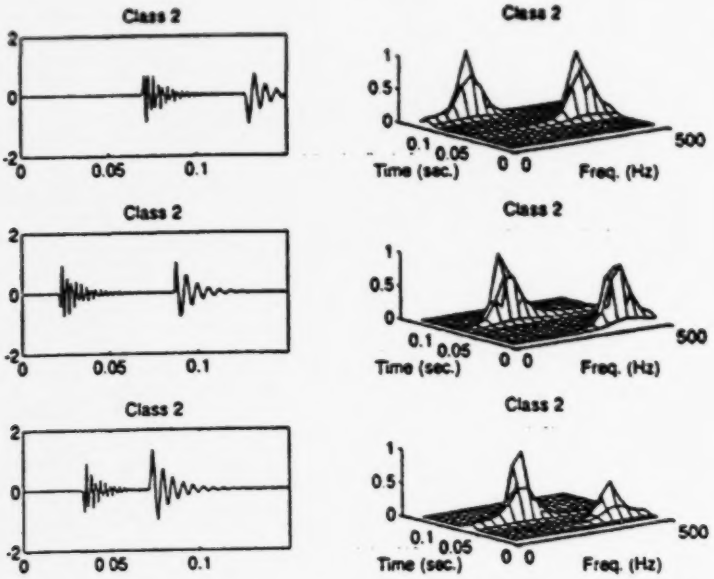

Figure 3: Noise–free Class 2 signals from the Three Tone Data Set with their respective spectrograms.

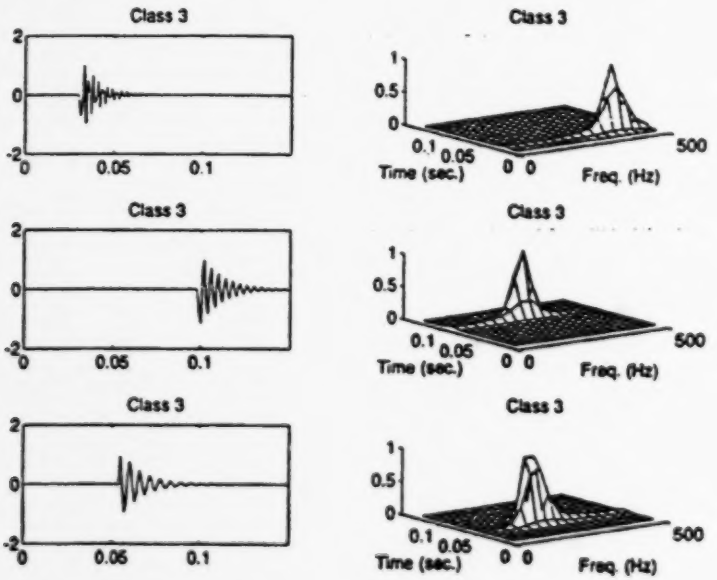

Figure 4: Noise–free Class 3 signals from the Three Tone Data Set with their respective spectrograms.

The Signal to Noise Ratio (SNR) was computed as

$$\text{SNR} = 10 \log \left[\frac{\sum_{k=1}^{k=152} (S(k) - n(k))^2}{\sum_{k=1}^{k=152} n^2(k)} \right] \text{(dB)}. \qquad (6)$$

The SNRs for the training and test set varied over a range of –20.4 dB to –2.6 dB. Figure 5 shows a typical signal and its associated spectrogram for each of the three signal classes.

Figures 6, 7, and 8 show samples of noisy time signals and their spectrograms for Classes 1, 2, and 3 respectively. These spectrograms are typical of the patterns which make up the test and training set.

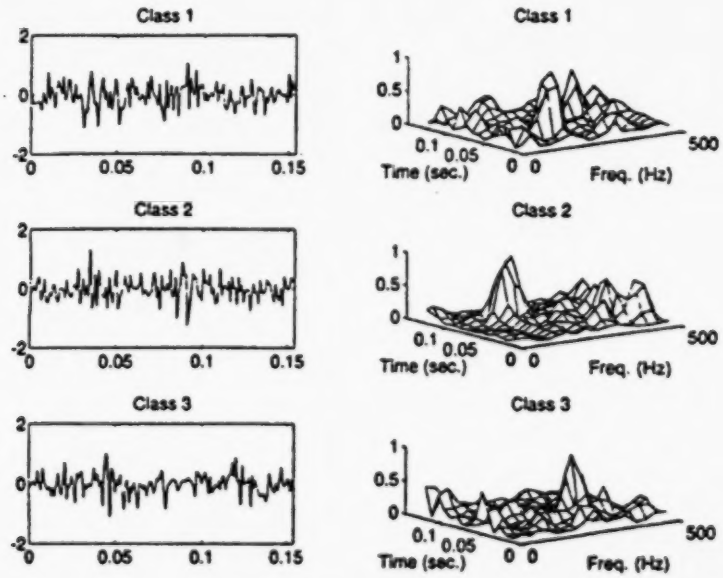

Figure 5: Sample signals from the Three Tone Data Set. For Classes 1, 2, and 3 the SNRs are –10.4, –8.0, and –17.0 dB respectively.

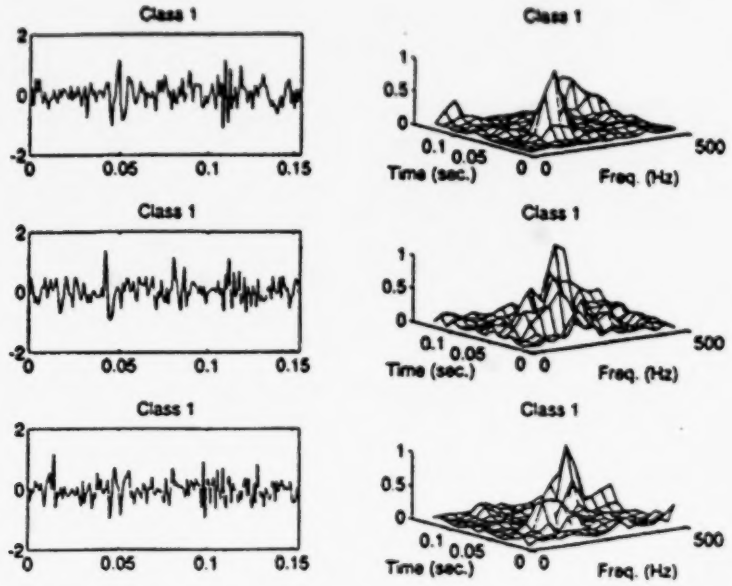

Figure 6: Class 1 signals from the Three Tone Data Set and their spectrograms. From top to bottom, the SNRs are –8.5, –8.8, and –8.5 dB.

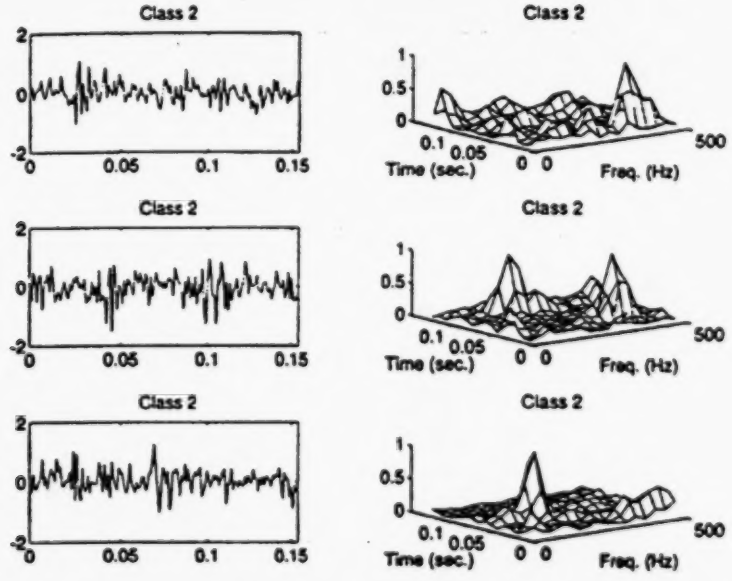

Figure 7: Class 2 signals from the Three Tone Data Set and their spectrograms. From top to bottom, the SNRs are –7.9, –11.2, and –9.1 dB.

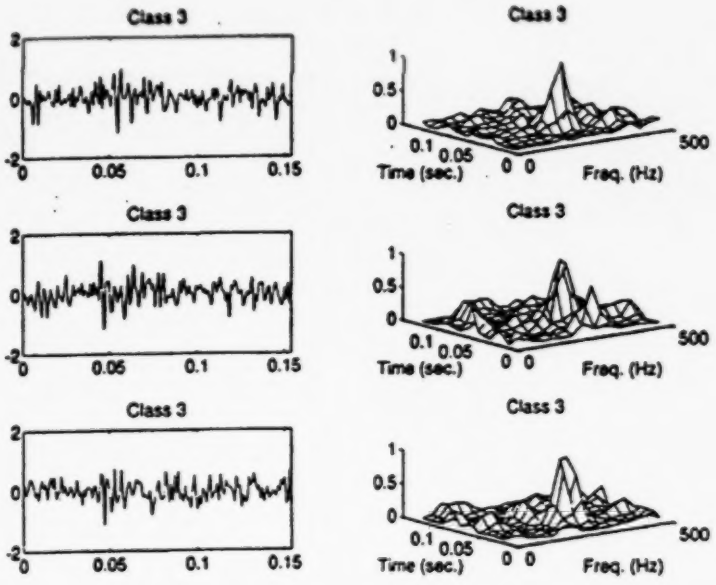

Figure 8: Class 3 signals from the Three Tone Data Set and their spectrograms. From top to bottom, the SNRs are –16.5, –17.1, and –15.9 dB.

3.2 Simulated DARPA Data Set (SDDS)

As previously mentioned, SDDS was based on a data set produced by DARPA as part of an Artificial Neural Network initiative meant to evaluate the effectiveness of different neural network architectures. SDDS was artificially generated with a special random number generator used to simulate ocean ambient noise[3]. Some notable differences exist between SDDS and the original DARPA data set (DDS), these are

- SDDS contains a noise only class while in some references [2 and 3] DDS does not,
- SDDS contains 1 extra signal class, a linear FM chirp,
- the signals of SDDS were sampled at a rate of 10 kHz, whereas the original DARPA data set was sampled at a rate of 24.576 kHz,
- the duration of classes D, E, and F have been changed, and
- the separation between the two pulses of class B has been changed.

The inclusion of a noise only class increases the problem difficulty level since the ambient noise spectrum will overlap into that of the signal classes. The FM chirp class was added

3. Using Webster's [8] notation we selected the following values needed to generate the ocean ambient noise: $a = 0.6$, $\beta_2(y) = 3$, $\beta_2(x) = 4.5$, and $n = 0.85$.

to include a signal type whose frequency signature evolved with time. The duration of the signals in classes D, E, and F was changed from 100 ms., 1 sec., and 8 sec. respectively to 60 ms. for each of the three classes to increase the similarity between classes, again making the classification problem more difficult than the original DDS.

Each sampled signal consists of 1024 points, corresponding to a 102.4 ms. snapshot for a sampling rate of 10 kHz. The eight classes of SDDS are briefly described as:

Class A: Broadband pulse, 15 ms. duration
Class B: Two broadband pulses (39 ms. separation), 4 ms. duration
Class C: 3 kHz tonal, 10 ms. duration
Class D: 3 kHz tonal, 60 ms. duration
Class E: 150 Hz tonal, 60 ms. duration
Class F: 250 Hz tonal, 60 ms. duration
Class G: linear FM chirp, 60 ms. duration
Class N: ocean noise only

Because we require the use of random variables to generate classes A, B and N and determine the offset of each signal, we will define *N(mean, variance, k)* as a *k* point sequence of independent and identically distributed (iid) Gaussian random variables with the specified mean and variance. We also define *U([min,max], k)* as a *k* point sequence of iid uniformly distributed random variables over the range *[min,max]*.

3.2.1 Class Descriptions

The following is an explanation of how each of the eight classes was generated. The SDDS was created using Matlab, Version 4.1 running on a Sun Sparc station. For all class descriptions, the sample index, k, ranges between 1 and 1024.

Class A

Defined as a broad band pulse of 15 ms. duration, Class A is modelled as a *N(0,1,450)* sequence multiplied by a gaussian envelope.

$$S_1(k) = \begin{cases} \exp\left[-\frac{1}{2}\left(\frac{k-(512+\epsilon)}{75}\right)^2\right] N(0,1,450) + n(k) & 288+\epsilon \le k \le 737+\epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (7)$$

The mean of the gaussian envelope is $(512+\epsilon)$ where,

$$\epsilon = U([-160, 160], 1), \quad (8)$$

and its standard deviation is 75. We define the *significant* portion of the signal $S_1(k)$ to be the 150 samples (15 ms.) within one standard deviation of the mean. The 1024 point noise sequence, $n(k)$ is generated using the ocean noise model described in [8].

Class B

Defined as two broadband pulses with 39 ms. peak separation and 4 ms. width, Class B is modelled as two *N(0,1,120)* sequences, each multiplied by a gaussian envelope.

$$S_2(k) = \begin{cases} \exp\left[-\frac{1}{2}\left(\frac{k-(317+\epsilon)}{20}\right)^2\right] N(0,1,120) + n(k) & 259+\epsilon \le k \le 378+\epsilon \\ \exp\left[-\frac{1}{2}\left(\frac{k-(707+\epsilon)}{20}\right)^2\right] N(0,1,120) + n(k) & 647+\epsilon \le k \le 766+\epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (9)$$

The two pulse centers are located at $(317+\epsilon)$ and $(707+\epsilon)$, where the random variable, ϵ is described by equation (8). The peaks of both pulses are separated by a constant of 39 ms. The standard deviation for each envelope is 20 samples, which means that the significant portion of each pulse is 4 ms. (or 40 samples).

Therefore, both pulses are centered at $(317+\epsilon)$ and $(707+\epsilon)$, a peak separation of 39 ms. The standard deviation for each envelope is 20 samples, therefore the total number of samples within one standard deviation of each pulse is 4 ms. or 40 samples.

Class C

A 3 kHz tone of 10 ms. duration, Class C is modelled as

$$S_3(k) = \begin{cases} \sin(2\pi(3000)kT_s) + n(k) & 463 + \epsilon \leq k \leq 562 + \epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (10)$$

where ϵ is defined as in Equation (8) and the sampling period T_s is 0.1 ms.

Class D

A 3 kHz tone of 60 ms. duration, Class D is modelled as

$$S_4(k) = \begin{cases} \sin(2\pi(3000)kT_s) + n(k) & 213 + \epsilon \leq k \leq 812 + \epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (11)$$

where ϵ is defined as in Equation (8).

Class E

A 150 Hz tone of 60 ms. duration, Class E is modelled as

$$S_5(k) = \begin{cases} \sin(2\pi(150)kT_s) + n(k) & 213 + \epsilon \leq k \leq 812 + \epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (12)$$

where ϵ is defined as in Equation (8).

Class F

A 250 Hz tone of 60 ms. duration, Class F is modelled as

$$S_6(k) = \begin{cases} \sin(2\pi(250)kT_s) + n(k) & 213 + \epsilon \leq k \leq 812 + \epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (13)$$

where ϵ is defined as in Equation (8).

Class G

A linear FM tone of 60 ms. duration, class G is modelled as

$$S_7(k) = \begin{cases} \mathrm{Re}[e^{j2\pi f_1 kT_s + j2\pi f_2 (kT)^2}] + n(k) & 213 + \epsilon \le k \le 812 + \epsilon \\ n(k) & \textit{elsewhere} \end{cases} \quad (14)$$

where ϵ is defined as in Equation (8).

Class N

Simulated ambient ocean noise, Class N is modelled as

$$S_8(k) = n(k) \qquad 213 + \epsilon \le k \le 812 + \epsilon \quad (15)$$

Figures 9 and 10 show typical time domain signals for all signal classes for low and high values of SNR, respectively. The SNRs are calculated as described below in section 3.2.2.

3.2.2 SNR Computation

The Signal to Noise Ratio was computed as

$$SNR = 10 \log\left[\frac{Signal\ \ Power}{Noise\ \ Power}\right] \quad (16)$$

where the *Signal Power* is computed as the sum of squared components over the significant portion of the time signal, which varies from class to class. The *Noise Power* is computed as the sum of squared components over the same time frame.

3.2.3 Spectrogram Generation

The STFT was computed using a 256 point Hamming window with 75% overlap. The resulting spectrogram is a 13 by 128 (time x freq) matrix. The bin resolution for the frequency axis is half the sampling rate divided by the number of frequency bins, 5000/128 = 39.1 Hz. The bin resolution for the time axis is 64 samples or 64/10000 = 6.4 ms. Since each signal class has a maximum possible offset of 160 samples in either direction, the location of the signal in the spectrogram varies between $\pm 160/64 = \pm 2.5$ time frames. In order to reduce the required computation time for training to a reasonable length, it was necessary to re-

duce the number of frequency bins from 128 to 16. A new frequency scale was created, the SDDS scale was based on the Mel scale which has been used extensively in speech processing [4] since it is based on the characteristics of the human cochlea. See figure 11 for an illustration of the Mel and SDDS scale resolution.

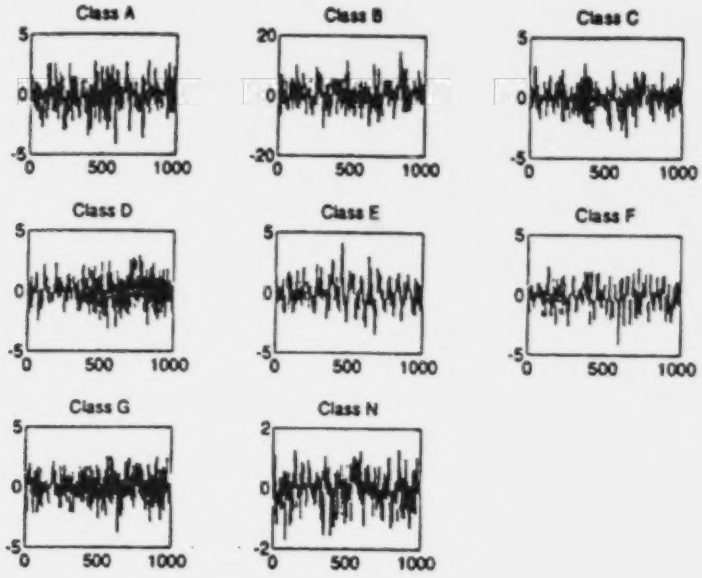

Figure 9: SDDS time domain signals for SNRs of 0 dB. These signals are typical of those included in SDDS.

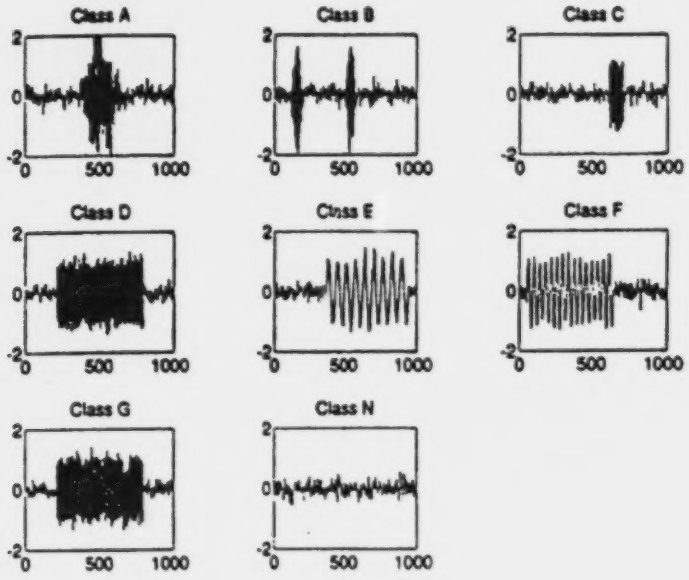

Figure 10: SDDS time domain signals for high SNRs ranging between 33 and 48 dB. These signals are shown for illustrative purposes only; the SNRs of the signals included in the SDDS training and testing sets were on the order of 0 dB.

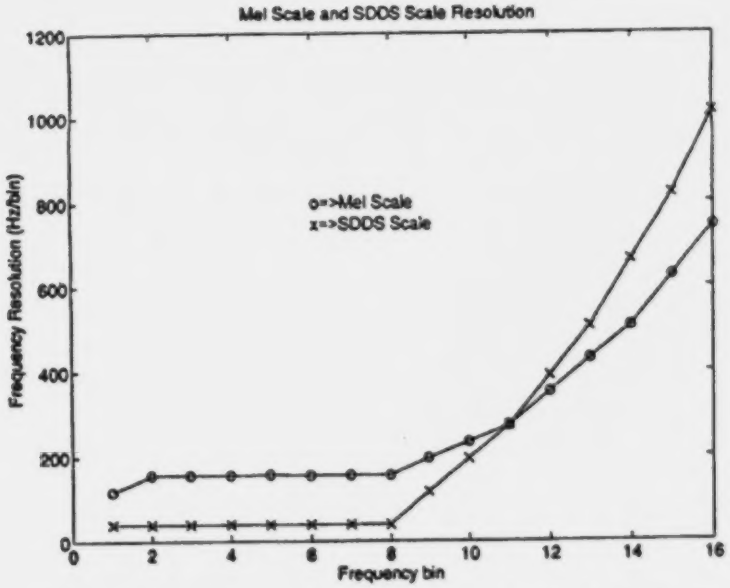

Figure 11: Illustration of frequency resolution for Mel and SDDS scales.

From the SDDS scale we can observe that the first 8 frequency bins are transferred directly and the next 8 frequency bins have values assigned by a summation of the energy over an increasing number of frequency bins. Figure 12 shows samples of spectrograms for each signal class with a 0 dB SNR and with the frequency bins compressed according to the SDDS scale.

3.2.4 Training and Test Set Characteristics

Each data set has a total of 800 patterns each having 13x16=208 components. The sequence of pattern classes is a sequence of iid discrete–valued random variables uniformly distributed over the integers 0 and 7. The SNR for the patterns in both data sets was set to zero. The spectrograms were normalized by their individual peak values so that the elements of the normalized spectrograms have values between 0 and 1. Representative spectrograms of the test and training set are shown in Figure 12.

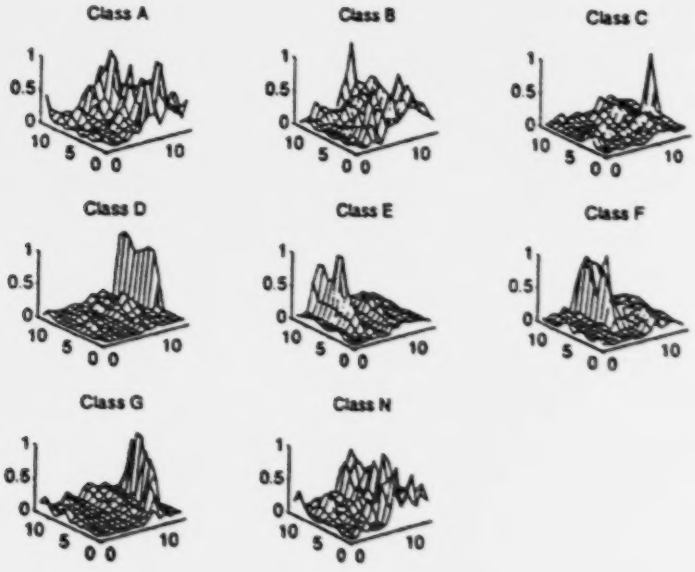

Figure 12: SDDS spectrograms with 0 dB SNR and reduced according to SDDS scale.

3.3 Multipath Data Set (MPDS)

In order to fully investigate the properties and capabilities of the FIR Neural Network structure in a multipath environment, a data set which incorporates effects of a multipath channel has been generated. This data set is made up of six signal classes, three of which are modelled on the Morlet Wavelet [9], two on the Gaussian Weighted Linear Frequency Modulated (GWLFM) Wavelet [9], and one class will consist of noise only. The signals are generated by convolving the wavelets with the impulse response of the multipath channel. Intra–class variations are produced when random fluctuations are introduced in the model of the multipath channel. Noise is added to the resulting signal and a spectrogram representation of this data is then obtained.

3.3.1 Morlet Wavelet

The Morlet Wavelet (a gaussian weighted tone) is defined as follows:

$$s(k) \equiv s(kT_s) = Re \ [\ e^{-\alpha(kT_s)^2} e^{j2\pi fkT_s} \ ] \tag{17}$$

where *Re* indicates the Real part of the complex equation, T_s the sampling period and k the sample index. Each Wavelet (Morlet and GWLFM) consists of 128 points sampled at 4 kHz ($T_s = 0.25$ ms). Currently, the three Morlet signal classes have the following values of α and f.

Table 2: Morlet Wavelet parameters for Classes 1, 3, and 5.

Class	α	f (Hz)
1	15000	50
3	60000	210
5	105000	370

The noise–free time domain representation of these signal classes is shown in Figure 13.

3.3.2 GWLFM Wavelet

The Gaussian Weighted Linear FM Wavelet is defined as follows:

$$s(k) \equiv s(kT_s) = Re[e^{-\alpha(kT_s)^2} e^{j2\pi f_1 kT_s + j2\pi f_2 (kT_s)^2}] \quad (18)$$

Again, each wavelet is made up of 128 points sampled at 4 kHz. Table 3 shows the current values of α, f_1, and f_2 for classes 2 and 4.

Table 3: GWLFM Wavelet parameters for Classes 2 and 4.

Class	α	f_1 (Hz)	f_2 (Hz)
2	37500	130	1500
4	82500	290	1500

The time domain representation of classes 2 and 4 is also shown in Figure 13.

3.3.3 Modelling the Multipath Channel

The impulse response for the nominal model of the simulated multipath channel is shown in Figure 14; its duration is 96.25 ms. (which corresponds to 385 samples at a sampling rate of 4 kHz); it is comprised of 4 gaussian pulses centered at 5, 15, 35 and 65 ms. (20, 60, 140 and 260 samples) with pulse widths (as measured by the portion of the pulse within 3 standard deviations of the pulse center) of 3, 6, 12, and 24 ms., and with heights of $e^{-a(k_c T_s - 0.005)}$ for all four gaussian pulses, where k_c denotes the sample number corresponding to the center of the pulse, $a = 25$, and $T_s = 0.25$ ms. Variations in the multipath channel are introduced by allowing variability in the mean, standard deviation and height of each gaussian pulse. The changes (from nominal) in the mean, standard deviation and height of each gaussian pulse are modelled as random variables with normal distributions having means of 0 and standard deviations equal to 15% of the corresponding nominal value. Therefore, each signal in the training and testing sets will have been subjected to a slightly different multipath channel. The nominal impulse response of the multipath channel and several random samples are shown in Figure 14.

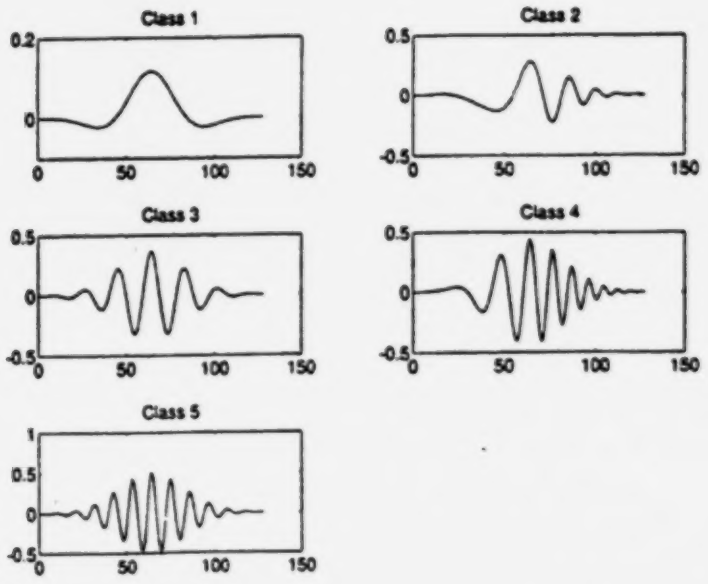

Figure 13: Illustration of Morlet wavelets (Classes 1, 3, and 5) and GWLFM wavelets (Classes 2 and 4).

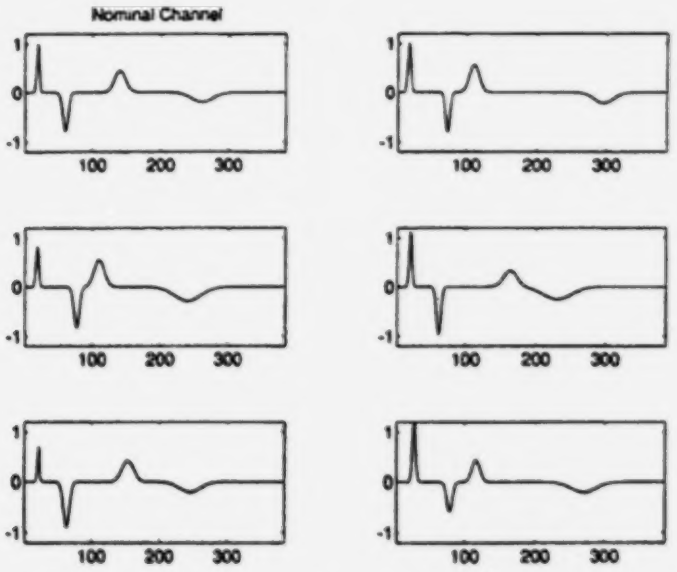

Figure 14: Nominal impulse response of multipath channel and several random examples.

3.3.4 Addition of Noise

The convolution results in a 512 point signal sampled at 4 kHz. Gaussian random noise with zero mean and unit variance is added to the signal that has passed through the multipath channel. The Signal to Noise Ratio (SNR) is computed as 10 times the log of the ratio of Signal Power to Noise Power.

$$SNR = 10 \log\left(\frac{Signal \;\; Power}{Noise \;\; Power}\right) \tag{19}$$

The Signal Power is approximated as the power contained in the first lobe of the convolved signal; it is computed as the sum of squares of the sequence which results from convolving the wavelet with the first pulse of the nominal impulse response of the multipath channel. The Noise Power is computed as the sum of the squared elements of the noise vector over the same number of points as the Signal Power. The SNR for the signals in the test and training sets ranges between +5 dB and –25 dB.

3.3.5 Generation of Spectrograms

The STFT is computed with a 128 point Hamming window and an overlap of 75%. The spectrogram of the multipath signal (512 samples) results in an array of 13 time bins by 64 frequency bins. The bin resolution for the frequency axis is 2000/64 = 31.25 Hz and the resolution for the time axis is 32 samples or 32/4000 = 8 msec. Since all the frequencies of interest lie below 500 Hz only the first 16 frequency components are retained for presentation to the FIR Neural Network. Figures 15 and 16 show the multipath signals and their spectrograms without the addition of noise while Figures 17 and 18 show the same for various SNRs. Figure 18 is an example of the patterns which are presented to the FIR network. In order to facilitate a continuous mode operation, the spectrograms are no longer normalized by their individual maximum values. Instead all the spectrograms are normalized by the same factor. This factor was computed as the mean of the peak values of the spectrograms from a large sample set.

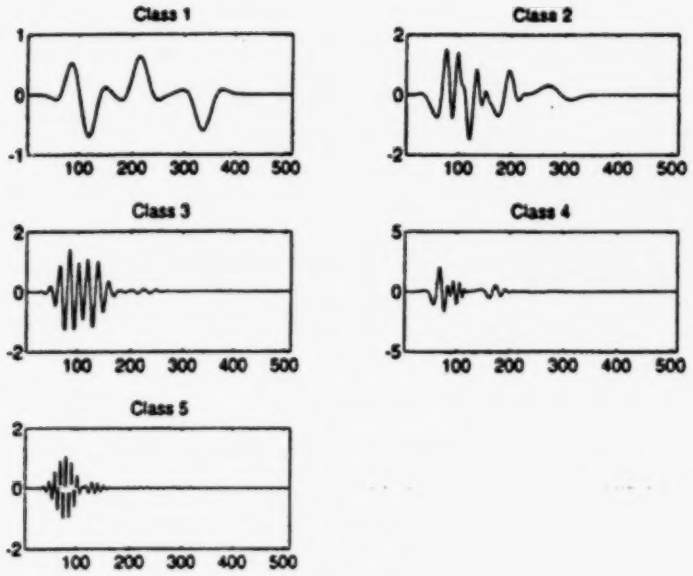

Figure 15: Signals resulting from the convolution of wavelets with the nominal impulse response of multipath channel for classes 1–5.

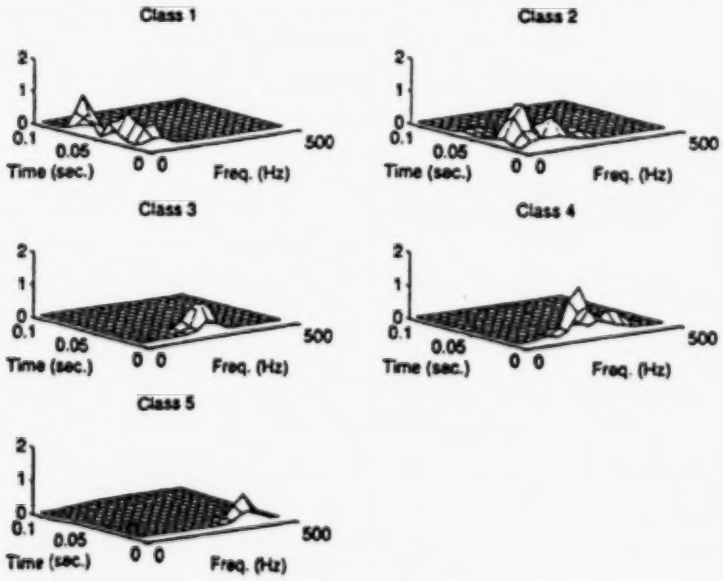

Figure 16: Spectrogram representation of signals resulting from the convolution of wavelets with impulse response of multipath channel for classes 1–5.

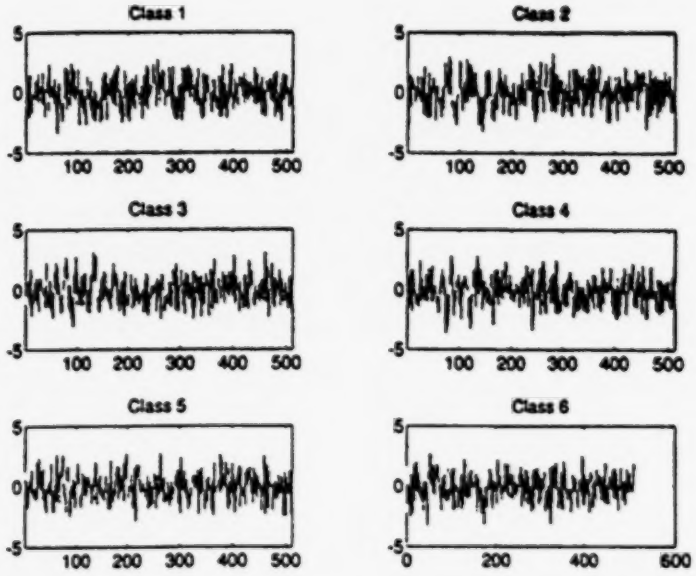

Figure 17: Signals resulting from the convolution of wavelets with impulse response of multipath channel with added noise for classes 1–5 (SNRs of –19.1, –5.0, –3.7, –0.1, and –2.9 dB respectively).

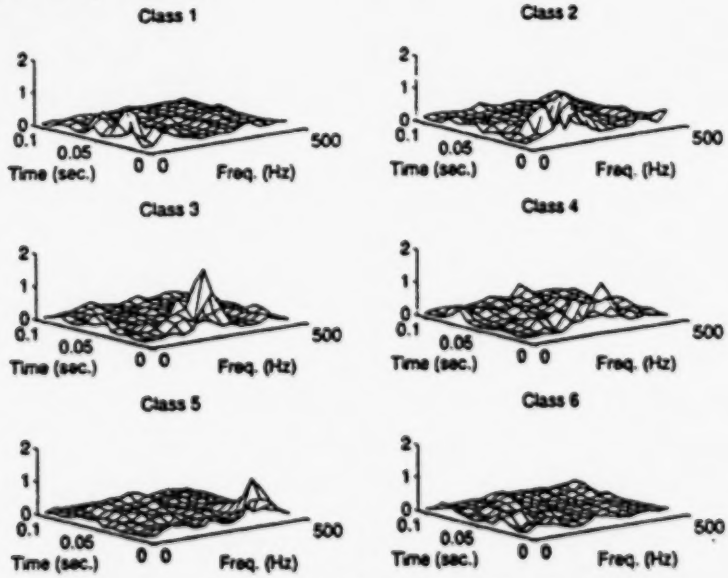

Figure 18: Spectrogram representation of signals resulting from the convolution of wavelets with impulse response of multipath channel with added noise for classes 1–5 (SNRs of –19.1, –5.0, –3.7, –0.1, and –2.9 dB respectively).

4 Weight Initialization and Update Frequency

The software allows for numerous options in initializing and updating the weights of the FIR neural network. Several experiments have been conducted to determine the effects of various options on the speed of learning and the resulting classification performance.

4.1 Weight Initialization

The weights of the FIR neural network can be initialized by two different methods. First, the user can specify a weight file that was previously saved, in which case the network will instantly assume the same performance it had when the weights were saved. Secondly, the weights can be initialized to random values uniformly distributed between plus and minus max_init_wt, where the value of max_init_wt is specified by the user in the network parameter file. Also specified by the user in the network parameter file is a seed for the random number generator.

The classification rate of the network after training until a steady state has been reached has been found to be insensitive to the value of the seed for the random number generator. This characteristic may indicate that local minima are not a serious problem for the classification tasks and FIR architectures that we have considered.

However, the range of the initial weight values does have an effect on the steady state classification rate. This is explained by the phenomena known as "premature saturation". Lee *et al.* [13] have shown that premature saturation may result when the weights are initialized to large values.

Figure 19 shows the classification rate for three different values of maximum initial weight, 0.1, 0.5, and 1.0. This experiment was conducted using the SDDS, a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture, a uniform learning rate (*i.e.*, all the layers with processing units have the same learning rate) of 0.001, flat desired responses of $+1$ for the desired node and -1 for the others, and impulsive error scaling meaning that only the error on the T^{th} iteration is backpropagated through the network, where T is the length of the pattern sequences (*i.e.*, the number of time frames per spectrogram).

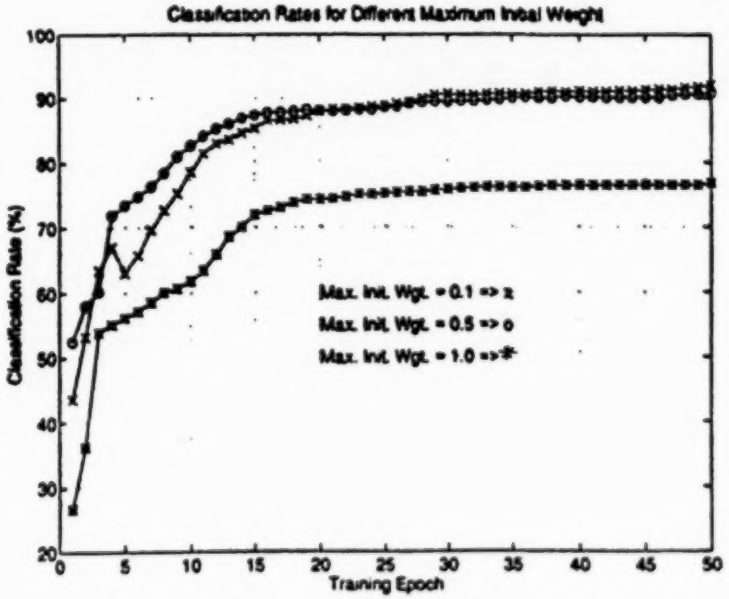
Classification Rates for Different Maximum Initial Weight
Classification Rate (%)
Training Epoch
Max. Init. Wgt. = 0.1 => x
Max. Init. Wgt. = 0.5 => o
Max. Init. Wgt. = 1.0 => *

Figure 19: Classification rates for the simulated DARPA test set using different values of maximum initial weights and a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture.

As seen in Figure 19, the trial corresponding to the smallest value of max_init_wt had the best steady state classification rate on the SDDS test set, while the trial corresponding to the largest value of max_init_wt had the worst steady state classification rate. In general, we have obtained good results with max_init_wt equal to 0.1.

4.2 Weight Update Frequency

The user is capable of changing the frequency of weight updates by making changes to the user options file. The software allows for weight updates to occur every iteration, every pattern or any integer multiple number of patterns. For all weight update frequencies, the weight update equation can be written as

$$w_{qmn}^{(\ell)}(i+1) = w_{qmn}^{(\ell)}(i) - \mu \sum_{p=1}^{P} \sum_{k=1}^{K} \delta_q^{(\ell)}(k,p) a_m^{(\ell-1)}(k-n,p). \qquad (20)$$

The vector of weights associated with the FIR filter connecting node m in layer ℓ to node q in layer $\ell+1$ is defined as $w_{qm}^{(\ell)}$. The individual weights of this filter are denoted by

$w_{qmn}^{(\ell)}$, for $0 \leq n \leq M^{(\ell+1)}$. The variables p and k denote the k^{th} input vector of the p^{th} input sequence. For a more detailed explanation refer to Section 3.2 (Notation and Description of FIR Neural Networks) in [10]. We can see that for the iteration–wise update scheme, P and K both take on the value 1. For anything other than iteration–wise weight updates, K will become equal to the number of iterations performed per pattern ($2T$–1), and P will become the integer value that specifies the number of patterns processed between weight updates.

Several experiments have been conducted using the TTDS and multiple network architectures to determine whether a smaller weight update frequency can result in improved learning characteristics. It was hoped that by performing weight updates less frequently the many minor weight corrections which often occur in opposite directions would be replaced by a bigger weight change oriented in a more "correct" direction in weight space.

Unfortunately, we did not find any significant improvement in learning characteristics when switching from iteration–wise updates to pattern–wise or multiple pattern–wise updates. We caution the user that as the update frequency decreases, the learning factor must also be decreased. This is due to the fact that the accumulated weight changes get bigger as the number of patterns between weight updates gets bigger (see equation 20).

Figure 20 shows the classification rate for the Three Tone test and training sets, described in section 3.1, for weight updates occurring every iteration, every pattern, every 5 patterns, and every 20 patterns. All results are based on a learning rate of 0.001 and an FIR[(16:0) x (5:8) x (3:7) x (3:0)] architecture. Note that the performance of the network becomes more erratic as the time between weight updates increases. This erratic behavior is similar to the behavior which is observed when using a learning factor that is too big. This reinforces the conclusion that the learning factor should be decreased as the frequency of weight updates decreases. However, even when decreasing the learning factor for the multiple pattern wise updates, no improvements were observed. For this reason, most of our subsequent experiments were conducted with iteration–wise updates.

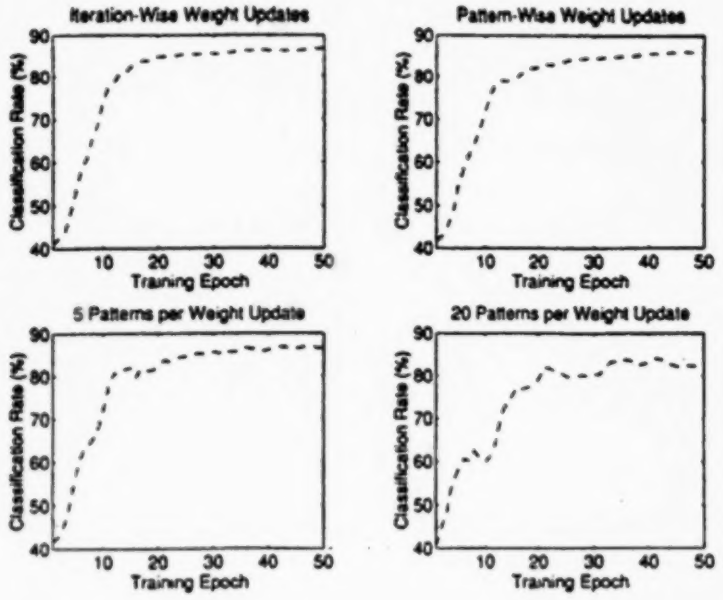

Figure 20: Classification rates of Three Tone test and training set using a FIR[(16:0) x (5:8) x (3:7) x (3:0)] architecture and various weight update frequencies. The dotted and dashed curves represent the training and test sets respectively.

4.3 Summary

Thus, to summarize we note that when initializing the weights:

- the FIR network architecture demonstrates an insensitivity to initial conditions for equivalent maximum initial weights, which may imply that local minima are not a serious problem,
- the user should choose a small value for the maximum initial weight (say 0.1) to speed learning and improve performance,

When choosing the weight update frequency:

- increased time between weight updates requires smaller learning rates, and
- no improvement in learning or performance was observed while using non–iteration–wise weight update techniques.

5 Desired Response and Error Scaling

Supervised training algorithms such as Temporal Backpropagation require that a desired–response sequence be supplied for each input sequence in the training set. During training, an error vector is calculated at each iteration as the difference between the desired outputs and actual outputs of the network. The resulting error sequence can be scaled in time (prior to backpropagation through the network) in order to place more or less emphasis on errors which occur at different stages in the process of clocking an input pattern through the network. Several strategies for selecting desired response and error–scaling sequences are discussed in [10]. The software currently allows for three different types of error scaling sequences and two types of desired response sequences. Adding to the existing library of desired response and error scaling sequences is easily accomplished by making simple additions to fir_nn.c. To choose from the available options, the user simply has to assign the appropriate values to ScaleErrsFlag and desired_resp_flag contained in the user options file. The available options for desired response and error scaling are explained in the software user's manual but repeating that information would be helpful before proceeding.

If the user sets desired_resp_flag equal to 0, the desired response for the output node corresponding to the true pattern class will be set to 1 for the length of time it takes to clock the input pattern (a multidimensional sequence of length T) in and out of the network (a total of $2T-1$ iterations), this will be referred to as a *flat desired response*. If the user sets the desired_resp_flag equal to 1, the desired response for the same output node will be a gaussian–shaped function (of the iteration number) – it will have a value close to 0 when the first vector of the input pattern is clocked into the network, it will reach its peak value of 1 at the time when the pattern fills the network, and will taper back down to 0 as the pattern is clocked out of the network. This type of desired response will be referred to as a *gaussian desired response*. All the elements of the desired response sequences for the output nodes not associated with the true class of the given input sequence are set equal to –1.

If the user sets ScaleErrsFlag equal to 0, equal emphasis is placed on all errors (*i.e.*, no error scaling occurs). If the user sets ScaleErrsFlag equal to 1, all emphasis is placed on

the error corresponding to the instant of time when the input sequence fills the network, this will be referred to as *impulsive error scaling*. Finally, if the user sets ScaleErrsFlag equal to 2, most of the emphasis is placed on the instant of time when the input sequence fills the network; the emphasis tapers down to zero (following a gaussian–shape curve) for those instants of time when only one vector of the input sequence has been clocked into, or has yet to be clocked out of, the network. This will be referred to as *gaussian error scaling*.

5.1 Results

Several experiments have been performed using many different configurations of desired response and error scaling. Though many combinations have been tried, some of which are no longer available as options in the FIR neural network software package, we cannot claim that an exhaustive search examining all possible permutations of desired response and error scaling has been performed. Five experiments using the Simulated Darpa Data Set were performed, for each of two different architectures. Architecture #1 was FIR[(16:0) x (12:1) x (8:5) x (8:6)] and architecture #2 was FIR[(16:0) x (12:4) x (8:6) x (8:2)]. For all the experiments using different combinations of desired response and error scaling, architecture #2 outperformed architecture #1 in terms of classification rate. The filter order distribution for architecture #2 was more heavily weighted towards the earlier layers than architecture #1, this leads to a significant difference in the number of adaptable weights in the two networks, therefore allowing architecture #2 to store considerably more knowledge within its structure.[4] Since the purpose of this section is to demonstrate the effects of desired response and error scaling, only the results concerning architecture #2 will be presented. Table 4 summarizes the five experiments performed using architecture #2 and the Simulated Darpa Data Set.[5]

4. For a four layer network with nonadaptable weights in the final layer, the number of adaptable weights is computed as $\{ (n^{(0)} \times n^{(1)} \times (M^{(1)}+1)) + (n^{(1)} \times n^{(2)} \times (M^{(2)}+1)) + n^{(1)} + n^{(2)} \}$.

5. Experiment 5 uses the type of gaussian desired response as described in previous section.

Table 4: Experiments performed on the Simulated Darpa Data Set using architecture #1.

Experiment	Desired Response	Error Scaling
1	true class: gaussian 0 to 1 others: gaussian 0 to –1	none
2	flat	impulsive
3	flat	none
4	flat	gaussian shape
5	true class: gaussian 0 to 1 others: flat at –1	none

A uniform learning rate of 0.001 and iteration–wise weight updates were chosen for all five experiments. A maximum initial weight of 0.1 was used for experiments 2, 3, and 4 and a maximum initial weight of 1.0 for experiments 1 and 5. Experiments 1 and 5 were performed before a lower maximum initial weight was found to yield a higher classification rate; unfortunately these experiments have not yet been repeated with a maximum initial weight of ± 0.1. The training and test set classifications rate for all five experiments are shown in Figure 21. Table 5 summarizes the classification rate after 50 epochs along with the *80% crossing point* (*i.e.*, the number of training epochs required to achieve a classification rate of 80% on the test set) which gives an indication of learning speed.[6] As we can see Experiment #2 yielded the best classification rate; as measured by its 80% crossing point, it was slightly slower than experiments 1, 3, and 4.

Table 5: Classification rate on test set after 50 epochs and number of training epochs required to achieve a classification rate of 80% on the test set.

Experiment	Classification Rate After 50 Epochs	80% Crossing Point
1	89.4%	9
2	93.25%	10
3	88.75%	8
4	88.4%	3
5	87.6%	24

6. The results in Table 5 are very preliminary as they are based on a single trial of each experiment. At this point, we do not know if the 80% crossing points would be consistent from one trial to the next.

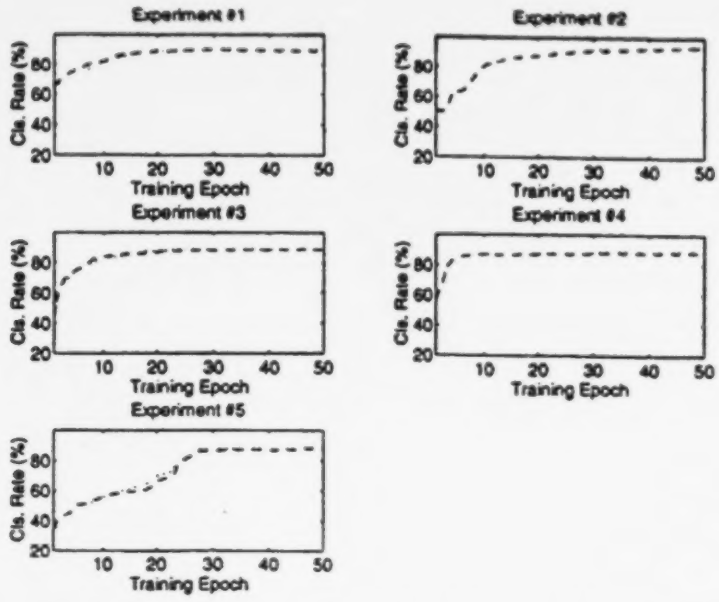

Figure 21: Classification rates (as a function of training epoch) on SDDS test and training sets using architecture #1. Dotted and dashed line represent the training and test sets respectively.

5.2 Summary

The two options, desired response and error scaling go hand–in–hand because the choice of one type of desired response or error scaling will eliminate or increase the effect of the other. As we can see from Figure 21 and Table 5 the impulsive error scaling experiment had the best results in terms of classification rate. The choice of impulsive error scaling should eliminate any effect of choosing different desired responses as long as all the desired response functions have the same value on the T^{th} iteration, which has been true for all our experiments to date. On the other hand if no error scaling is chosen then the effect of the combined options is solely dependant on the choice of desired response. For the experiments performed, impulsive error scaling has yielded the best results, thereby making the choice of desired response irrelevant. Another advantage of impulsive error scaling is that training requires fewer operations per pattern. When using impulsive error scaling, only the forward pass need be carried out during the first T–1 iterations and only the backward pass need be carried out during the last T iterations. On the other hand, the other error–scaling sequences require both forward pass and backward pass to be con-

ducted at each of the $2T-1$ iterations required to clock the input sequence in and out of the network. An additional $T-1$ iterations of backward pass are required to accumulate the weight updates generated by the error vector at iteration $2T-1$. At present, the software uses all $3T-2$ iterations for each pattern regardless of the chosen error–sequence.

6 Time of Occurrence

Other than the classification of underwater acoustic transient signals, another area of interest is to determine the signal time of occurrence. Specifically, in addition to classification of a spectrogram (according to transient type), we would like to identify the alignment of the transient within the spectrogram. To this end, we have examined the actual output of the nodes in the final or penultimate layers of the network. As explained in the user's manual, the penultimate layer nodes are not fully connected to the final layer nodes. Instead a filter connection exists only between corresponding nodes in the penultimate and final layers. Thus, the output of each final layer node is simply a time average of the last $(M^{(L)}+1)$ outputs of the corresponding penultimate layer node.

6.1 Method of Approach

To examine the feasibility of determining the transient alignment from the outputs of the penultimate and/or final layer nodes, we did the following.

1. A FIR[(16:0) x (12:4) x (8:6) x (8:2)] network was trained and tested on the Simulated DARPA Data Set, learning options included a uniform learning rate of 0.001, iteration wise–weight updates, maximum initial weights of ± 0.1, flat desired responses and impulsive error scaling. The classification rate and the mean square error for the test and training set are plotted as the number of training epochs in Figures 22 and 23. After training for 50 epochs, the network achieved a classification rate of 93.25% on the test set. Table 6 shows a confusion matrix for the simulated DARPA test set after training for 50 epochs, this table explains how the network is performing on a per–class basis. As we can see, Classes B and N account for the majority of incorrect classifications with Classes C, D, E, F, and G having a 100% classification rate. After training for 50 epochs all the weights of the network were saved for future reference.
2. Three new pattern sets were then created using the eight classes from the simulated DARPA data sets. Each pattern set contained only eight spectrograms, with each class appearing once, sequenced in increasing numerical order (0 to 7). Each of the signal classes was generated as explained in Section 3.2.1, except that the offset, ϵ, was no longer determined by a random variable, instead all the patterns in each data set had the same offset, with each pattern set having a different offset. The pattern sets darpa1, darpa2, and darpa3 have offsets ϵ_1, ϵ_2, and ϵ_3 which assume the values –160, 0, and +160 samples respectively (corresponding to offsets of –16, 0, and +16 ms. or –2.5, 0, +2.5 time frames respectively).

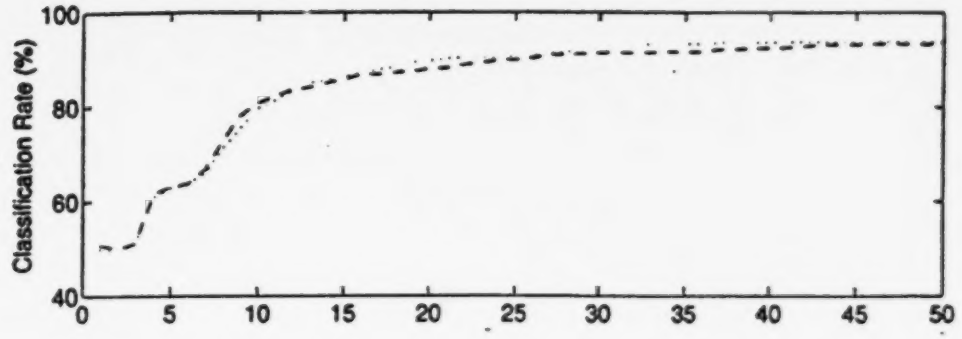

Figure 22: Classification rate for the Simulated DARPA test and training sets using a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture. Dotted and dashed curves represent the training and test set respectively.

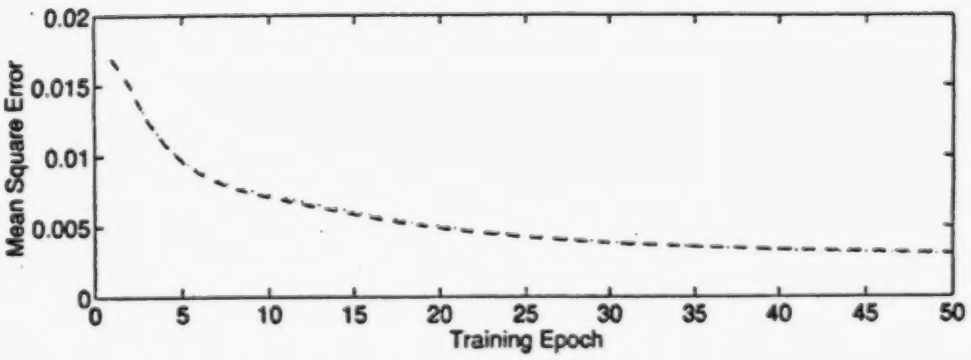

Figure 23: Mean square error curves for the simulated DARPA test and training sets using a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture. Dotted and dashed curves represent the training and test set respectively.

Table 6: Confusion matrix for simulated DARPA test set after training for 50 epochs.

		Chosen Class							
		A	B	C	D	E	F	G	N
True Class	A	102	1	0	0	0	0	0	0
	B	2	54	0	0	0	0	0	39
	C	0	0	98	0	0	0	0	0
	D	0	0	0	119	0	0	0	0
	E	0	0	0	0	100	0	0	0
	F	0	0	0	0	0	98	0	0
	G	0	0	0	0	0	0	97	0
	N	2	10	0	0	0	0	0	78

3. The weights of the network were then set equal to the saved weights from the trained network in step 1. The three pattern files: darpa1, darpa2, and darpa3 were clocked in and out of the network during which the outputs from all the nodes in the penultimate and final layers were saved. Figures 24 and 25 show the output for all the nodes of the final and penultimate layers for pattern set darpa2, which has offset 0, over the $2T-1=25$ iterations required to clock in and out the eight spectrograms (each spectrogram consists of $T=13$ time frames). Since we know the sequence of classes in this pattern set, 0 to 7, we can zoom in on the output of each node for the $2T-1$ iterations during which the pattern representative of the node's class is presented to the network. Figures 26 through 33 show the output of the final layer nodes for the 25 iterations of interest for each node; note that the output resulting from each of the three pattern sets (darpa1, darpa2, and darpa3) are superimposed so as to allow the reader to compare the peak locations to the offsets (all patterns were classified correctly in darpa3, while classes N and B were classified incorrectly from darpa1 and darpa2 respectively). Only the output from the final layer is presented as the output from this layer has proven easier to work with and has yielded better results.

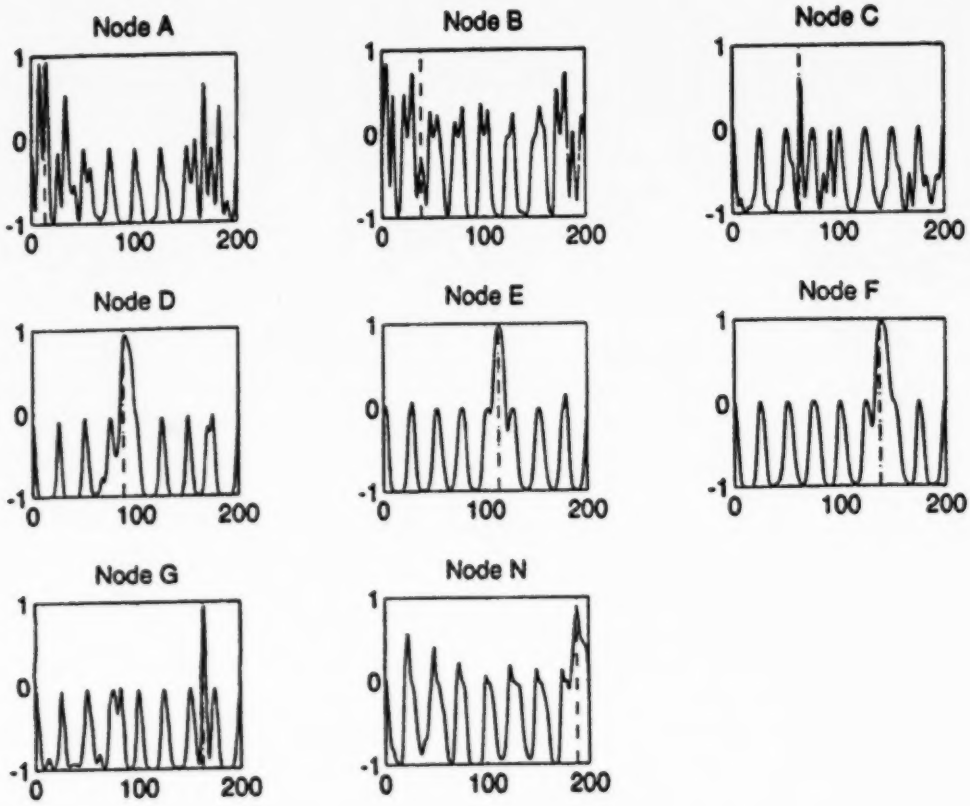

Figure 24: Final layer output for nodes 0 through 7 (corresponding to classes A through N as indicated) of trained FIR network using pattern set darpa2, with offset 0. The dashed vertical lines indicate the time at which we expect the output of the node to go high.

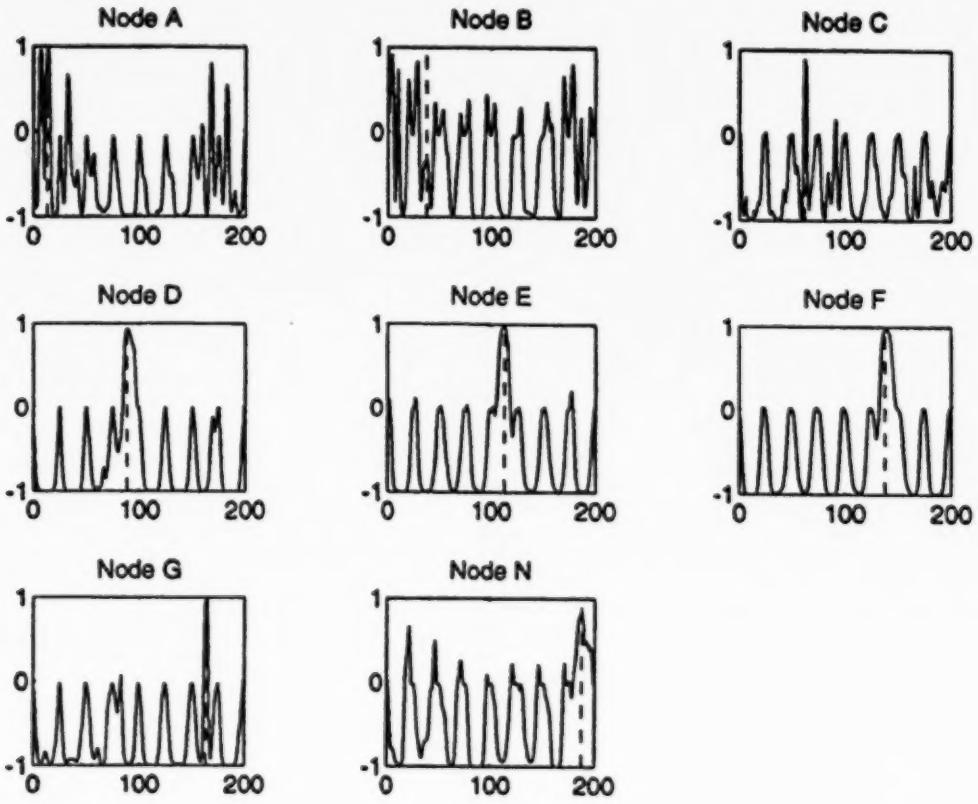

Figure 25: Penultimate layer output for node 0 through 7 (corresponding to classes A through N as indicated) of trained FIR network using pattern set darpa2, with offset 0. The dashed vertical lines indicate the time at which we expect the output of the node to go high.

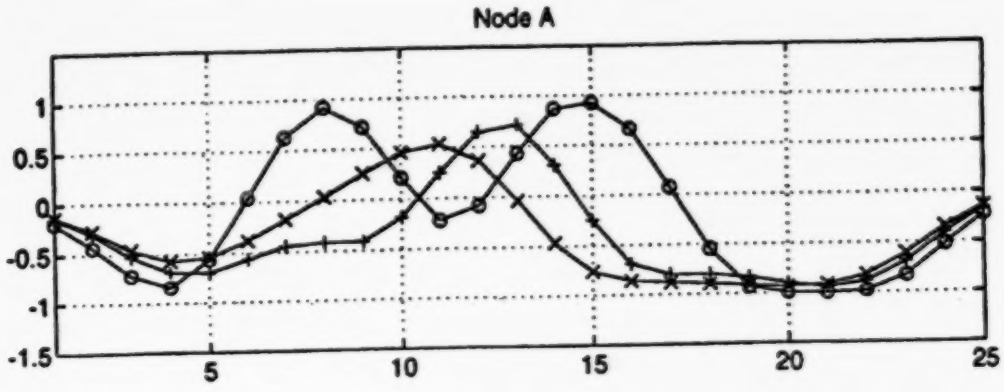

Figure 26: Final layer output of node A for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

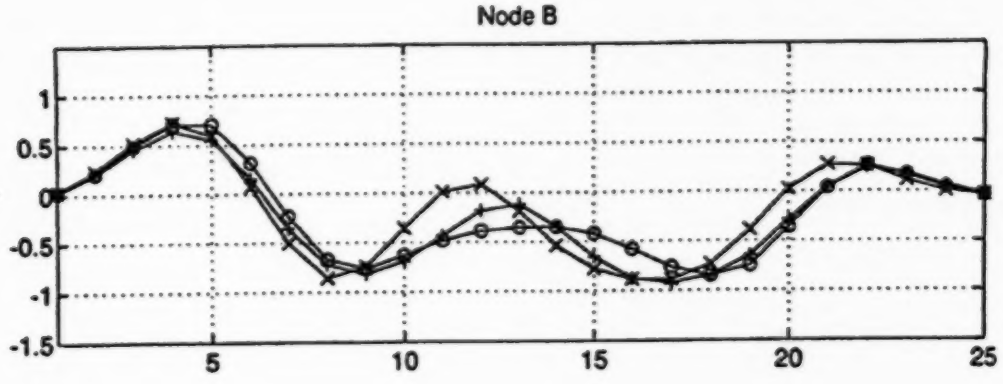

Figure 27: Final layer output of node B for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

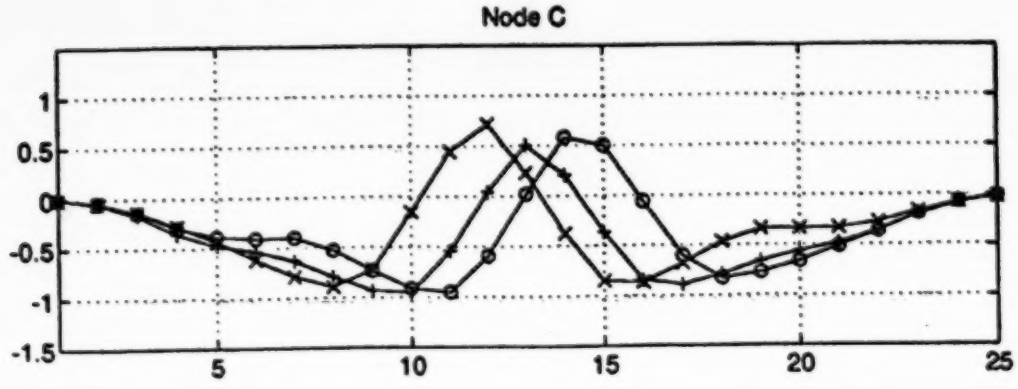

Figure 28: Final layer output of node C for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

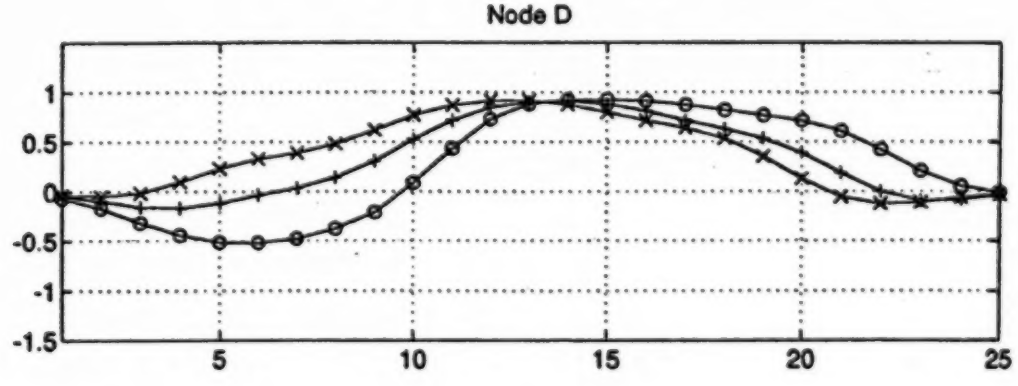

Figure 29: Final layer output of node D for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

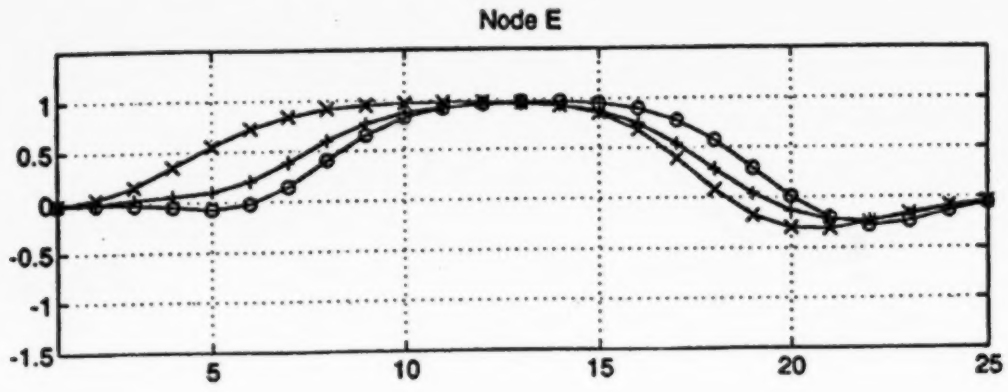

Figure 30: Final layer output of node E for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

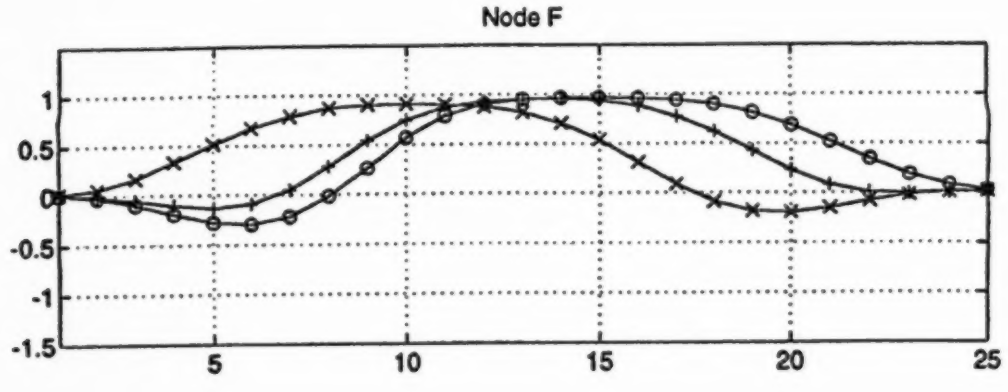

Figure 31: Final layer output of node F for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

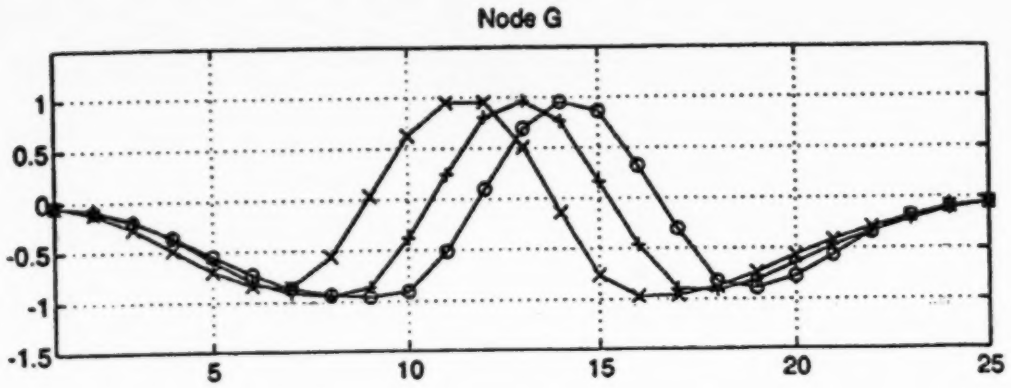

Figure 32: Final layer output of node G for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

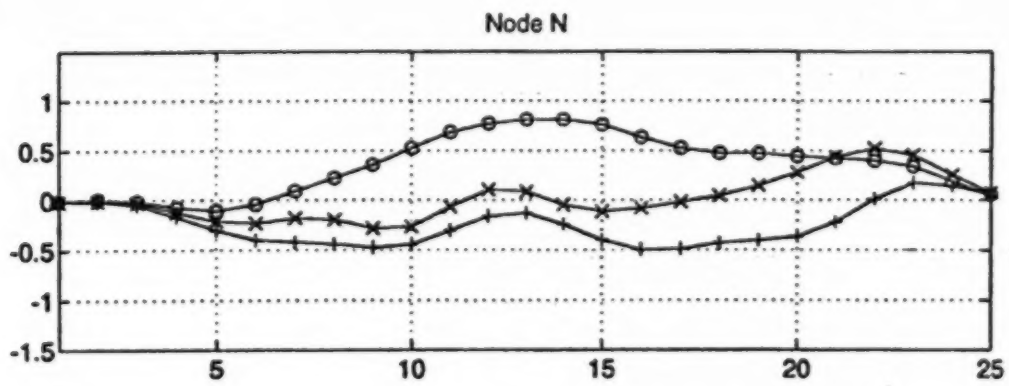

Figure 33: Final layer output of node N for pattern sets darpa1 ("x"), darpa2 ("+"), and darpa3 ("o") from the trained FIR network.

Table 7: Peak location for all the classes in darpa1, darpa2, and darpa3.

	darpa1	darpa2	darpa3
A	11	13	15
B	4	2	4
C	12	13	14
D	13	14	15
E	11	13	14
F	10	14	15
G	12	13	14
N	22	23	14

6.2 Observations for Determining Transient Alignment

Table 7 summarizes the location of the peaks (in terms of iteration number) of the true class output for each pattern in the three darpa sets. From Table 7 and Figures 26 through 33 we make the following observations:

- if we look at the interval of time over which each true node should be active, iterations 1 to 25, the peaks of the 3 different patterns of the same class occur in the same order as the offsets of those 3 different pattens (*i.e.*, the output resulting from the spectrogram which has the significant portion of signal occurring first in time will typically peak first). Exceptions occur when the pattern has been classified incorrectly.
- for most spectrogram classes having offset, $\epsilon_1 = -160$ samples, the peak will occur at or before the 12^{th} iteration,
- for most spectrogram classes having offset, $\epsilon_2 = 0$ samples, the peak will occur on the 13^{th} sample, and
- for most spectrogram classes having offset, $\epsilon_3 = +160$ samples, the peak will occur on or after the 14^{th} sample.

6.3 Suggested Procedure for Determining Transient Alignment

From the observations noted above in section 6.2 we have constructed the following rules from which to determine the transient's offset.

if peak occurs at or before iteration 12
offset $= \epsilon_1$
else if peak occurs at iteration 13
offset $= \epsilon_2$
else if peak occurs after or at iteration 14
offset $= \epsilon_3$

To test our formulation a new simulated DARPA pattern set was created. This pattern set, darpa4, contains 200 patterns (approximately equal representation of all 8 classes) with

the offset uniformly distributed over the range of discrete values (–160, 0, +160). The pattern set was clocked in and out of the network, keeping the same weights as before, and the output of all the final layer nodes was saved. We then scanned the nodes on the 13^{th} iteration to determine the network's classification of each spectrogram. Next we scanned to determine the iteration at which the peak value occurred and then applied the rules above to determine the offset of each transient. This value was then compared with the actual value of the offset, which was stored when the data set was generated. The results are shown in Table 8. The results of Table 8 are based on scanning a total of 25 iterations for the peak output. If instead, we repeat the above procedure but replace the location of the peak output over iterations 1 to 25 with the location of the peak output over iterations 10 to 15, we get the results shown in Table 9.

For the procedure used to generate the results in Table 8, the offset was correctly estimated 77% of the time; the frequency of correct estimation went up to 80.5% for the procedure used to generate the results of Table 9. Of the patterns which were correctly classified, 84.9% had their offsets correctly estimated.

Table 8: Offset estimation for pattern classes in darpa4 using output from all 25 iterations.

Class	Number of Correct Offset Estimations	Number of Incorrect Offset Estimations
A	24	3
B	5	19
C	28	0
D	26	0
E	24	6
F	15	5
G	25	0
N	7	13
totals	154	46

Table 9: Offset estimation for pattern classes in darpa4 based on the output for iterations 10–15.

Class	Number of Correct Offset Estimations	Number of Incorrect Offset Estimations
A	24	3
B	10	14
C	28	0
D	26	0
E	24	6
F	15	5
G	25	0
N	9	11
totals	161	39

We should note that the rules which were used for the offset estimation were very crude. In reality it makes sense to develop a set of rules for each pattern class and we believe that the estimation accuracy would increase if this were done. It may also be worth considering other alternatives to the peak location for determining offset; for example, as an alternative we might use the mean location of a cluster of consecutive outputs above some threshold. It would also be interesting to examine the effect of various desired–response and error–scaling sequences on the accuracy of offset estimation.

7 Comparison with Static Neural Networks

In order to gauge the performance of the FIR structure, we have compared the performance of the FIR neural network with that of a static neural network for the Three Tone Data Set and the Simulated DARPA Data Set. The results are discussed below.

7.1 Three Tone Data Set

The best performance obtained on the Three Tone test set using a FIR neural network had a FIR[(16:0) x (5:10) x (3:5) x (3:0)] architecture[7], a uniform learning rate of 0.001, maximum initial weights of ± 1.0, flat desired responses and no error scaling. The best performance obtained with the static network had an architecture of 256, 9, and 3 nodes in the first, second, and third layers respectively, a learning rate of 0.001 and maximum initial weights of ± 1.0. Figure 34 shows the classification rate for the Three Tone test set for both the static and FIR neural networks after training for 50 epochs. As we can see the FIR network classification rate after 50 epochs was 88.3% while that of the static network was 86.8%. The 80% crossing point occurred at epochs 12 and 19 for the FIR and static networks respectively. Tables 10 and 11 show the confusion matrices for the Three Tone test set for both networks after training for 50 epochs. The number of adaptable weights in the FIR network was 978, while the static network had a total of 2599.

Table 10: Confusion matrix for Three Tone test set after training for 50 epochs using a static neural network.

		Chosen Class		
		1	**2**	**3**
True Class	**1**	282	4	37
	2	1	309	25
	3	23	42	276

7. A total of five different architectures were tried, the four others were FIR[(16:0) x (5:5) x (3:5) x (3:5)], FIR[(16:0) x (5:8) x (3:7) x (3:0)], FIR[(16:0) x (5:12) x (3:3) x (3:0)], and FIR[(16:0) x (5:5) x (3:10) x (3:0)].

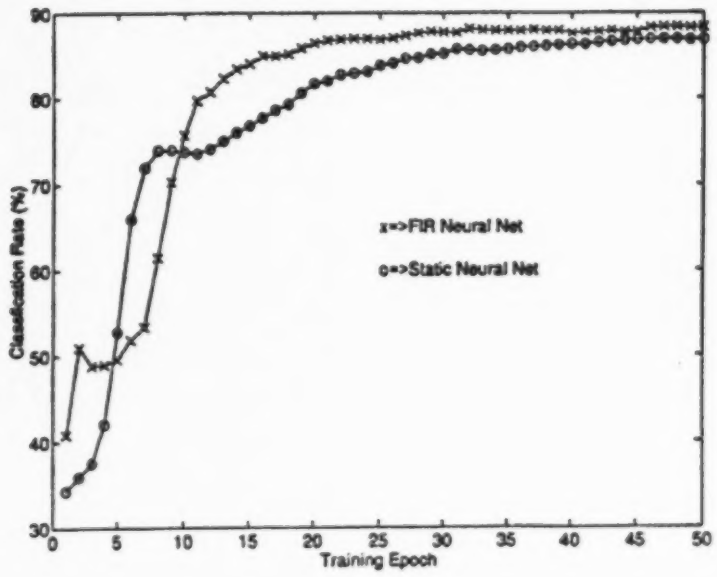

Figure 34: Classification rates for Three Tone test set using static and FIR neural networks.

Table 11: Confusion matrix for Three Tone test set after training for 50 epochs using a FIR neural network.

		Chosen Class		
		1	2	3
True Class	1	296	10	17
	2	4	313	18
	3	42	26	273

7.2 Simulated DARPA Data Set

The best performance obtained by the FIR neural network was a classification rate of 94.0% on the SDDS test set after training for 50 epochs. This performance was achieved using an FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture with a uniform learning rate of 0.002, iteration wise weight updates, maximum initial weights of ± 0.1, and flat desired responses with impulsive error scaling. Figure 35 shows the classification rate for the SDDS test and training sets.

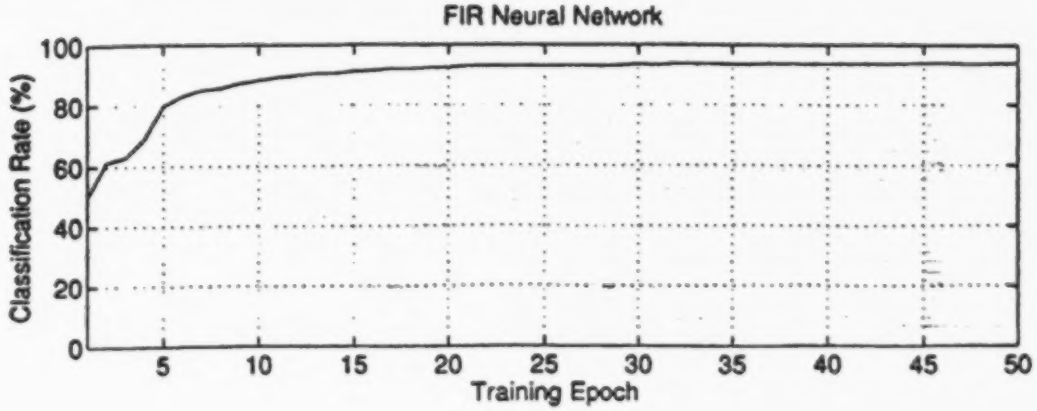

Figure 35: Classification rate as a function of the number of training epochs for the SDDS test set and training set using an FIR[(16:0) x (12:4) x (8:6) x (8:2)].

Table 12 shows the confusion matrix for the SDDS test set after training for 50 epochs.

Table 12: Confusion Matrix for the SDDS test set after training for 50 epochs.

		Chosen Class							
True Class		A	B	C	D	E	F	G	N
	A	101	1	0	0	0	0	0	1
	B	2	65	0	0	0	0	0	28
	C	0	0	98	0	0	0	0	0
	D	0	0	0	119	0	0	0	0
	E	0	0	0	0	100	0	0	0
	F	0	0	0	0	0	98	0	0
	G	0	0	0	0	0	0	97	0
	N	2	14	0	0	0	0	0	74

We then tested a static network with 208 inputs, 12 nodes in the hidden layer, and 8 nodes in the output layer. A learning rate of 0.0005 was used with the maximum initial weights

varying between ± 0.1. Larger values of learning rate were tried (0.002 and 0.001) but the classification rate as a function of training epoch was very erratic. The classification rate obtained on the test set is shown in Figure 36.

The confusion matrix for the SDDS test set after training for 200 epochs is shown in Table 13.

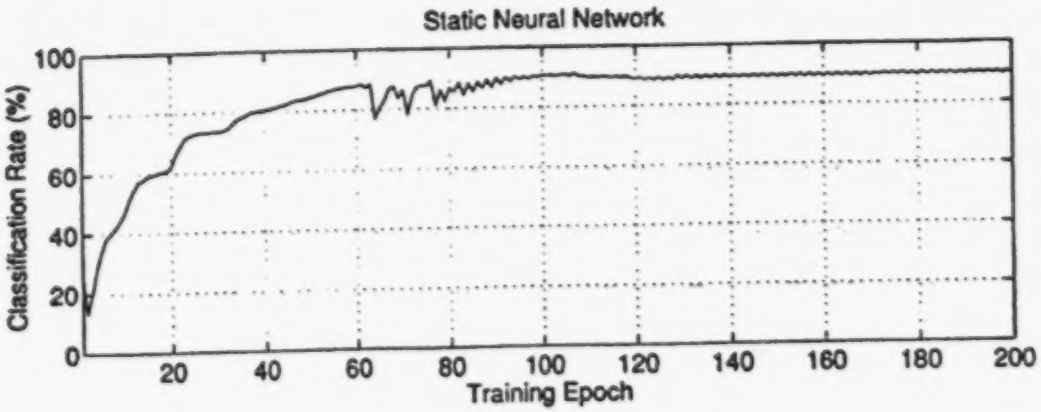

Figure 36: Best obtained classification rate on SDDS test set using static neural network.

Table 13: Confusion matrix for SDDS after training for 200 epochs on static feedforward multilayer network.

		Chosen Class							
		A	B	C	D	E	F	G	N
True Class	A	101	1	0	0	0	0	0	1
	B	4	16	0	0	0	0	0	75
	C	0	0	98	0	0	0	0	0
	D	0	0	0	119	0	0		0
	E	0	0	0	0	100	0	0	0
	F	0	0	0	0	0	98	0	0
	G	0	0	0	0	0	0	97	0
	N	0	0	0	0	0	0	0	86

The maximum classification rate achieved by the static neural network was 91.125%. We should also note that the FIR network converged to the correct solution much faster than the static network, the FIR network achieved an 80% classification rate after only 5 epochs whereas the static network required 39 epochs to achieve an 80% classification rate. Note that the learning factor for the FIR network was 0.002, whereas for the static network it was 0.0005. The static network had a total of 2612 adaptable weights and the FIR network only required 1652.

7.3 Summary

We should note that the FIR network outperformed the static network in terms of classification rate and time of convergence for both data sets. The FIR network also required significantly fewer adaptable weights than the static network.

8 Behavior in Multipath Environment

The objective of studying the use of FIR neural networks in a multipath environment is to determine how the performance of the network is affected by changes in the multipath channel (depth, temperature, etc...). As previously mentioned in section 3.3, we have generated a data set which attempts to incorporate the effects of intra–class variations due to randomly occurring changes in a multipath channel. Training and testing with this data set was performed using a FIR[(16:0) x (12:4) x (6:6) x (6:2)] architecture, uniform learning rate of 0.001, iteration–wise weight updates, maximum initial weights of ± 0.1 and flat desired responses with impulsive error scaling, the network achieved a classification rate and mean square error as depicted in Figures 37 and 38. After training for 50 epochs, a classification rate of 85% was achieved on the test set; the associated confusion matrix is shown in Table 15.

Table 14: Confusion matrix for multipath test set after training for 50 epochs using a FIR[(16:0) x (12:4) x (6:6) x (6:2)] architecture.

		Chosen Class					
		1	**2**	**3**	**4**	**5**	**6**
True Class	**1**	88	5	0	0	2	3
	2	2	108	1	0	1	0
	3	0	1	98	6	0	1
	4	0	2	15	66	2	5
	5	1	0	0	0	79	11
	6	8	2	4	9	9	71

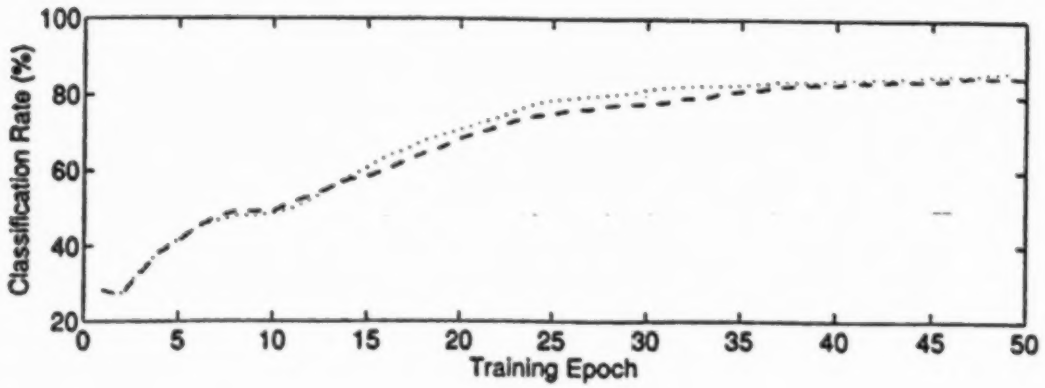

Figure 37: Classification rate for multipath test set using a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture. Dotted and dashed curves represent the training and test sets respectively.

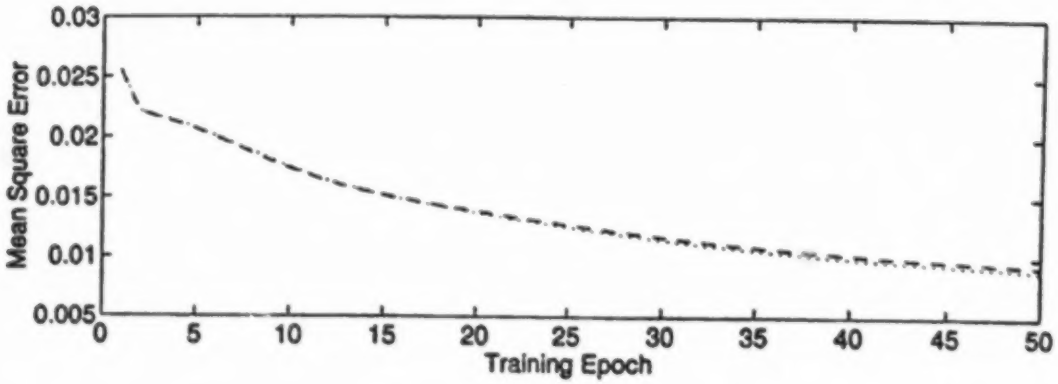

Figure 38: Mean square error for MPDS using a FIR[(16:0) x (12:4) x (8:6) x (8:2)] architecture. Dotted and dashed curves represent the training and test sets respectively.

9 Conclusions and Suggestions for Future Work

The FIR neural network architecture is well suited for temporal processing tasks which is the main reason that it was chosen for the purposes of this contract. The software which was developed to implement the FIR neural network package has many user options (as explained in the accompanying User's Manual [11]). Numerous experiments have been conducted on three simulated data sets in an effort to determine the effects of various user options and to assess the ability of the FIR neural network to classify acoustic transients. Many conclusions have been drawn based on our experimental results; however, we caution the reader that our results and conclusions should be treated as preliminary since time did not permit the vast amount of experimentation which would be necessary to thoroughly investigate the many options included in the FIR software package. We hope that our experimental results will help the reader to better understand the options available and we encourage further experimentation.

The details of the three data sets on which our experiments have been conducted are provided in chapter 3. The data sets are referred to as the Three Tone Data Set (TTDS), the Simulated DARPA Data Set (SDDS), and the MultiPath Data Set (MPDS). The patterns in these data sets are multidimensional sequences (spectrogram representations of the time domain signals). All data sets contain intra–class variations.

All experimental results were obtained using a four layer network (2 layers of adaptable weights). Such a network is capable of implementing an arbitrarily complex decision surface [12] and has been successfully applied to phoneme classification tasks [4]. The number of nodes in the input layer was always chosen equal to the dimension of the input pattern (*i.e.*, the number of frequency bins in the spectrogram); the number of nodes in the penultimate and final layers were always set equal to the number of pattern classes. The number of nodes in the remaining layers were determined through *a priori* knowledge and empirical observations. The filter order for the entire FIR structure was set equal to one less than the input sequence length (*i.e.*, the number of time frames per spectrogram), T. The filter order of the input layer was always set equal to zero; that of the final layer was typically set equal to the possible misalignment (in time frames) of the signal within the spectrogram. The filter order for the remaining layers was determined through *a priori*

knowledge and empirical observations. Some of our preliminary findings are summarized below.

A uniform learning rate of 0.001 was found to work well in most instances.

When initializing the weights to random values, a low value for the maximum initial weight should be chosen in order to prevent premature saturation [13], for most cases a value of 0.1 worked well. The classification rate has demonstrated to be insensitive to the seed used for the random number generator which initializes the weights, this may indicate that local minima are not a serious problem for FIR networks.

Performing weight updates every iteration was found to be the most effective weight update frequency; smaller learning factors should be used if pattern–wise or multiple pattern–wise updates are chosen.

The combination of desired–response and error–scaling sequences which has yielded the best classification results is a flat desired response sequence of $+1$ for the true class node and -1 for the other nodes, and an impulsive error–scaling sequence which places all emphasis on the error at the T^{th} iteration. This choice of error scaling allows for approximately a 33% reduction in the number of iterations performed during training (from $3T$–2 to $2T$–1) and reduces the number of subroutine calls to forward_pass and backward pass by half. Currently, the software does not implement the impulsive error–scaling sequence option any more efficiently than the others; it uses the same set of instructions regardless of the chosen error–scaling sequence. If implemented differently, a considerable reduction in training time could be realized.

The signal time of occurrence can be determined by closely observing the output of the final layer nodes. The location of the peak for the true class output node is a function of the offset or misalignment of the signal. The earlier the signal occurs within the spectrogram, the earlier the peak will occur on the true class node (if pattern is correctly classified). For tests using a 200 pattern SDDS and restricting the peak search between iterations 10–15, the offset was correctly determined 84.9% of the time when the patterns were correctly classified. This result was achieved using a set of very crude rules to determine

the offset; performance should improve with the inclusion of a set of class–specific rules to determine the signal offset.

The FIR neural network outperformed a static neural network on both the TTDS and the SDDS. The FIR network achieved classification rates of 88.3% and 94% on the TTDS and the SDDS respectively; the static network achieved classification rates of 86.8 and 91.125% on the same two data sets. Preliminary results indicate that the FIR structure converges faster to a higher steady state classification rate than the static neural network.

The signals included in the multipath data set were generated by simulating the propagation of various wavelets through a multipath channel; random changes were introduced in the model of the multipath channel at each use. Although we make no claims that our MultiPath data set is a realistic simulation, we note that the FIR neural network was able to achieve a classification rate of 85% on the MultiPath test set.

In summary, our results have been promising and we look forward to an opportunity to apply the FIR neural network to real oceanic signals.

9.1 Suggestions for Future Work

There are several areas of interest which remain to be explored in the future. Some suggestions are listed below.

9.2 Test on Actual Data

Although our pattern sets may approximate ocean acoustics, they can in no way encompass all the properties of such an environment, therefore, it would be extremely interesting to compare the FIR neural networks performance to that of other classification schemes in current use, human or not.

9.3 Continuous Mode Operation

At present, the FIR software package is set up to classify fixed length patterns (each pattern being a sequence of length T). In this mode of operation, the classification decision for each pattern is made based on the output of the network at iteration T when the sequence

fills the network. It would be interesting to examine what happens when the network operates in a continuous mode (i.e., concatenate all the test sequences into one long sequence to be continually input to the network); in this case, the outputs of the network would have to be continually monitored (rather than just checked at iteration T of a pattern presentation) for an indication of a pattern from a particular class. It is very likely that in this mode of operation, the optimal type of desired–response sequence and error–scaling sequence will be different as it will be important to suppress intermittent peaks which may presently occur when the signal is being clocked in and out of the network.

9.4 Experiment with other types of preprocessing techniques

Ghosh *et al.* [2] claim to have found that wavelet based coefficients were more effective features than spectral coefficients. It would be interesting to see if the performance of the FIR network improved using wavelet–based coefficients in place of our spectrograms.

9.5 Further experimentation with error–scaling and desired–response sequences

As mentioned in section 6.3, it would be interesting to investigate the effects of various desired–response and error–scaling sequences on the accuracy of offset estimation.

Also as mentioned in section 9.3, it is likely that for operation in a continuous mode, further experimentation will be required to determine the best combination of desired–response and error–scaling sequences. In continuous–mode operation, it will be important to suppress intermittent peaks; impulsive error scaling may not be as effective as other error–scaling techniques for suppressing intermittent peaks.

9.6 Further experimentation with architecture

To date, we have always used a 4–layer network (with two layers of adaptive parameters), it would be interesting to see if a 5–layer network might demonstrate any improvements. Also, we have done little experimentation with distribution of filter orders among layers. It would be interesting to do further experimentation to see how sensitive the performance is to the chosen filter–order distribution.

9.7 Further experimentation with learning factors and momentum

In designing the software, we have allowed for two features with which we have not had time to experiment: the possibility of different learning factors and momentum factors for

each layer. All experiments conducted to date have not made use of momentum and have used identical learning factors for all layers. It would be interesting to see the effect of these options on the speed of convergence and/or on the classification performance.

9.8 Optimize the FIRNN code for more efficient operation

There are no doubt numerous changes which could be made to the software to reduce training time. One such improvement is discussed at the end of section 5.2.

9.9 Hybrid System

Develop a hybrid system for the classification of underwater acoustic transients in which the FIR neural network would work in conjunction with one or more other types of classifiers (such as a Hidden Markov Model, or even another FIR neural network operating on a different set of features). The idea of a synergistic approach to the classification of acoustic transients is also discussed in [2] and [1].

9.10 Behavior in a Multipath environment

Depending on the operational scenario, it may be possible to use a FIR neural network or some other type of dynamic neural network to equalize the effects of a multipath channel. As mentioned in [10], Kechriotis *et al.* [14] have used a recurrent neural network for adaptive communication channel equalization.

References

[1] A. Kundu, G. C. Chen and C. E. Persons, "Transient Sonar Signal Classification Using Hidden Markov Models and Neural Nets," *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 1, pp. 87–99, January 1994.

[2] J. Ghosh, L. M. Deuser and S. D. Beck, "A Neural Network Based Hybrid System for Detection, Characterization, and Classification of Short Duration Oceanic Signals," *IEEE Journal of Oceanic Engineering*, Vol. 17, No. 4, pp. 351–363, October 1992.

[3] T. L. Hemminger and Yoh–Han Pao, "Detection and Classification of Underwater Acoustic Transients Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 5, No. 5, pp. 712–718, September 1994.

[4] A. Waibel, T. Hanazawa, G. Hinto, K. Shikano and K. J. Lang, "Phoneme Recognition Using Time–Delay Neural Networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 3, pp. 328–339, March 1989.

[5] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE Acoustics, Speech and Signal Processing Magazine*, pp. 4–22, April 1987.

[6] R. P. Gorman and T. J. Sejnowski, "Learned Classification of Sonar Targets Using a Massively Parallel Network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, No. 7, pp. 1125–1140, July 1988.

[7] R. P. Gorman and T. J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Networks*, Vol. 1, pp. 75–89, 1988.

[8] R. J. Webster, "A Random Number Generator for Ocean Noise Statistics," *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 1, pp. 134–137, January 1994.

[9] L. G. Weiss, "Wavelets and Wideband Correlation Processing", *IEEE Signal Processing Magazine*, Vol. 11, No. 1, pp. 13–32, January 1994.

[10] M. Stevenson, "Survey of Dynamic Neural Network Techniques with Applications to Temporal Processing Tasks", CR/97/458, April 1995.

[11] M. Stevenson, "FIR Neural Network Software Package User's Manual", CR/97/460, April 1995.

[12] D. R. Hush and B. G. Horne, "Progress in Supervised Neural Networks, What's New Since Lippmann?" *IEEE Signal Processing Magazine*, pp. 8–39, January 1993.

[13] Y. Lee, S. H. On, and M. W. Kim, "An Analysis of Premature Saturation in Back Propagation Learning," *Neural Networks*, Vol. 6, pp. 719–728, 1993.

[14] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using Recurrent Neural Networks for Adaptive Communication Channel Equalization," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, March, 1994.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)
University of New Brunswick
P.O. Box 440, Fredricton, N.B., Canada
E3B 5A3

2. SECURITY CLASSIFICATION (Overall security of the document including special warning terms if applicable.)
UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title.)
Classification of Simulated Sequences using a FIR Neural Network Software: Experimental Result

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)
ARSENAULT, Normand and STEVENSON, Maryhelen

5. DATE OF PUBLICATION (Month and year of publication of document.)
April 1995

6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)
61

6b. NO. OF REFS. (Total cited in document.)
14

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)
Contractor Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the reseach and development. Include the address.)
Defence Research Establishment Atlantic
P.O. Box 1012, Dartmouth, N.S., Canada B2Y 3Z7

9a. PROJECT OR GRANT NUMBER (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)
1cc

9b. CONTRACT NUMBER (If appropriate, the applicable number under which the document was written.)
W7707-3-2682/01-OSC

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number mrst be unique to this document.)

10b. OTHER DOCUMENT NUMBERS (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DREA Contractor Report 97/459

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; futher distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic annoucement of this document. This will normally correspond to the Document Availability (11). However, where futher distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

DCD03 2/06/87

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This report examines the experimental results achieved by the FIR Neural Network on data sets which simulate underwater acoustic transient signals. A software package to simulate FIR neural networks has been developed in the C programming language. The effects of several user controlled options will be given along with suggestions on how to choose these and the network parameters (number of layers, number of nodes, e.t.c.) which define the structure of the FIR neural network. Experiments have been conducted using three separate data sets (each data set contains the training set composed of a collection of spectrograms obtained from the Short Time Fourier Transform): a three tone data set (TTDS), the Simulated DARPA Data Set (SDDS), and a Multipath Data Set (MPDS). The effects of four user controlled options (weight initialization, weight-update frequency, choice of desired response, and error-scaling sequence) on the classification rate and learning speed is studied in order to determine the best configuration in which to set up the FIR software package. A suggested procedure to determine the transient time of occurrence is presented along with some results obtained on the SDDS. The performance of the FIR neural network is shown to be superior to that of static neural networks on both the TTDS and the SDDS. Finally, conclusions and suggestions for future work are given.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

passive sonar
neural networks
dynamic neural networks
finite impulse response
transient signals